PAPER PROCEEDINGS

THE BISHOP'S LODGE RESORT & SPA

Sandia National Laboratories

hosts the
Seventh PHANToM Users
Group Workshop
in beautiful Santa Fe, New Mexico

Sponsors:

Telemedicine & Advanced Technology Research Center
of the United States Army Medical Research and Materiel Command
Novint Technologies, Inc., Albuquerque, NM
SensAble Technologies, Inc., Woburn, MA

Table of Contents

A System for Accurate Collocation of Haptics and Graphics

Karl Reinig, Joshua Eskin, University of Colorado, Center for Human Simulations

I would conservatively estimate that in the last six years I have demonstrated haptic and graphic enabled virtual environments to over a thousand individuals. These individuals range from highly skilled surgeons to first graders. Having witnessed these interactions, I have come to the following conclusion: some people do better with Virtual Reality (VR) than others. Discussions of this variance usually focus on the computer familiarity of the user, which bodes well for the video game player. I would like to suggest the following hypothesis: The major difference in people's VR savvy is their varying ability to adapt to miscues in the virtual environments. If this is true, the following follow:

- The user's performance in a highly accurate virtual environment will not differ significantly from their performance in a real environment.
- There is a significant disparity across users between the correlation of the degradation of skills and degradation of the virtual environment.
- If standards are not developed and maintained, individuals not adept at adapting to miscues will be discriminated against as virtual environments make their way into the mainstream of training and testing.

Many systems have been developed to collocate the apparent graphic position of a virtual scene with a haptic workspace. The use of mirrors for this purpose is particularly popular, as they are inexpensive and have the natural effect of hiding the haptic from the viewer. However, the systems that I have experienced--including the one designed and used at the University of Colorado's Center for Human Simulation--allow for motion of the user's head without appropriately compensating for the implied change in the location and orientation of the virtual camera. The disparity between the true view point and the virtual viewpoint is a direct source of error in the collocation of the haptic and graphic scene.

For mirror systems, two methods come to mind for fixing this disparity: head tracking and fixing the position of the user's head. We have chosen to create a system that does the latter. In our latest system, the user views the virtual environment in much the same way that a person looks through a microscope. This fixes the eye position. There are many other factors that go into making an accurate virtual environment, including:

- calibration of the haptic display device
- calibration of the graphic display device
- graphic refresh rate
- graphic resolution
- proper stereo cues
- proper math

But all of the above are, to a large extent, under the control of the designer.

This presentation introduces a haptic and graphic workstation designed and built with the emphasis on presenting the user with an accurate virtual environment. We expect this system to be a useful display for many of our medical simulators. The more intriguing potential of the system lies in what it may teach us about people using virtual environments.

A 3-D Lasso Tool for Editing 3-D Objects: Implemented Using a Haptics Device

Marjorie Darrah
Institute for Scientific Research
Fairmont, WV 26554
mdarrah@isr.us

Alicia Kime
School of Science and Mathematics
Fairmont State College
Fairmont, WV 26554
akime@mail.fscwv.edu

Frances Van Scoy
Lane Department of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV 26505
fvanscoy@wvu.edu

Abstract

Many paint programs include a lasso editing tool that allows the user to capture an irregular portion of a two-dimensional figure. However, no corresponding tool is known to exist for three-dimensional editing. This paper describes a lasso tool, based on a convex hull algorithm, that has been simulated using the SensAble™ PHANTOM™ haptic device. The PHANTOM™ device is employed as a three-dimensional mouse to select a set of non-planar voxels, that is, three-dimensional pixels. The algorithm uses the selected voxels to define a convex hull. Once the voxels within the convex hull have been identified, they can be deleted, copied, or modified.

1 Introduction

This project addressed the problem of selecting a set of voxels (three-dimensional pixels) in a three-dimensional model for editing. The tools available for two-dimensional editing are rather flexible; many paint programs allow the mouse to sketch the boundary of a region, either freehand or using line segments to form a polygon. The user does not have to enclose the region in a circle, ellipse or square but may select a shape that is irregular. We sought to generalize the two dimensional idea of a lasso enclosing a irregular polygon to a lasso for three-dimensional sets of voxels.

Current packages on the market allow three-dimensional data to be edited in several ways including what are known as: the cookie-cutter (or extrusion) approach, the melon baller approach and the two-dimensional slicing approach [1]. The cookie-cutter approach allows an arbitrary shape to be drawn on a plane to enclose an area, then extends the shape downward orthogonal to the plane so that it selects a volume much like a cookie cutter presses through a large quantity of dough. The melon baller method uses a regular shape, such as sphere, cube, or rectangular prism to carve out sections of the three-dimensional space by using that shape. This allows greater control over the volume selected than the previous method, but it still yields a regular shaped region. In a third approach, two-dimensional slicing, a two-dimensional projection of the three-dimensional data set is edited repeatedly. The data set may be rotated to observe the data from different viewpoints, but the mouse is still editing in two dimensions.

The tool we created and simulated using the PHANTOM™ haptics device makes possible the freehand editing of a three-dimensional data set. Our prototype allows the user to specify a data set, containing points with (x,y,z) coordinates, which is then rendered as voxels in a three-dimensional haptic scene. To construct a convex hull, the haptics device is first used to select a set of three-dimensional voxels. When at least four non-planar voxels have been chosen, the program will construct a convex hull that encloses all points within the boundary of an irregular convex polyhedron defined by those voxels.

2 Mathematical Approach

To create a lasso tool that allows for the arbitrary selection of points and the enclosure of irregular three-dimensional volumes, we first explored the geometry of the problem. Just as the two-dimensional lasso tool uses a polygon to enclose the area to be edited, a three-dimensional lasso must define a polyhedron. A polyhedron is a region of space whose boundary is composed of flat polygon faces, any pair of which are either disjoint or meeting at edges and vertices.

It should be noted that we here are discussing and constructing a *convex* polyhedron. A set **S** of points is defined as convex if $\mathbf{x} \in \mathbf{S}$ and $\mathbf{y} \in \mathbf{S}$ implies that the line segment $\mathbf{xy} \subseteq \mathbf{S}$. A polyhedron that is constructed from the user-selected points is called a *convex hull* of the set. The following example may help visualize the concept of a convex hull. In a two-dimensional plane, the convex hull is the shape a string assumes when anchored to a nail at the lowest point with respect to y and wrapped

counterclockwise around nails pounded into the plane at each point. In three dimensions, the *boundary* of a convex hull is the shape taken by plastic wrap stretched tightly around all the points. More formally a definition of convex hull of a set **S** is the intersection of all convex sets that contain **S**.

Many algorithms exist for constructing both two and three-dimensional convex hulls. Graham's Algorithm [1], the fastest method for constructing a convex hull in two dimensions, has no obvious generalization to three dimensions. Among the best-known three-dimensional algorithms are Gift Wrapping (Chand & Kapur), Divide and Conquer (Preparata & Hong) and Incremental Algorithm (Seidel & Kallay). We chose to implement the Incremental Algorithm outlined by Joseph O'Rourke in the second edition of his, *Computational Geometry in C* [2].

In two dimensions the basic method of the Incremental Algorithm is to add points one at a time, constructing the hull of the first k points at each step and using that hull to incorporate the next point.

Algorithm: INCREMENTAL ALGORITHM

Let $H_2 \leftarrow \text{conv}\{p_0,p_1,p_2\}$.

For $k = 3$ to n-1 do

    $H_k \leftarrow \text{conv}\{H_{k-1} \cup p_k\}$

The algorithm begins with a triangle and considers whether the next point is inside or outside the area enclosed by that triangle. If the point is inside, then it is discarded; if it is outside, then the algorithm determines the two tangent lines from the new point to the triangle. The algorithm continues in this manner until the convex hull is fully defined.

This incremental algorithm, with time complexity of $O(n^2)$, can be generalized to three dimensions.

Algorithm: 3D INCREMENTAL ALGORITHM

Initialize H_3 to tetrahedron (p_0,p_1,p_2,p_3).

for i = 4, …, n-1 do

for each face f of H_{i-1} do

    Compute volume of tetrahedron determined by f and p_i.

    Mark f visible iff volume < 0.

    if no faces are visible

        then

            Discard p_i (it is inside H_{i-1}).

        else

            for each visible border edge e of H_{i-1} do

                Construct cone face determined by e and p_i.

            for each visible face f do

                Delete f.

            Update H_i.

The overall structure of the three-dimensional incremental algorithm is identical to that of the two-dimensional version. From a set of identified points, the algorithm chooses four non-planar points from which the initial hull is constructed. At the *ith* iteration, the algorithm computes the hull-in-progress by adding a new point. This action yields two possibilities: (1) if the new point is inside the existing hull, it is discarded; (2) if it is outside, the algorithm constructs a cone face determined by the new point and each border edge visible from that point. The process of defining the convex hull on the basis of a single new point p is visualized in Figures 1 - 3.

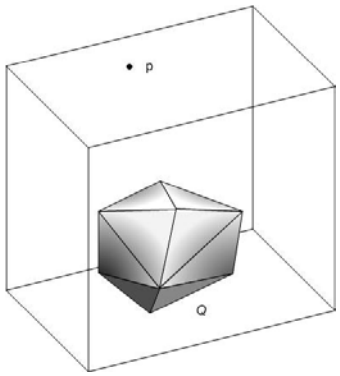

Figure 1: Convex hull Q and point p outside the hull

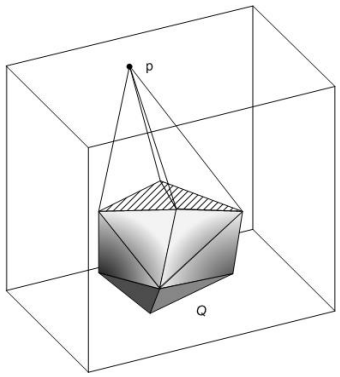

Figure 2: The striped faces are interior and marked for deletion when the hull is extended to include point p.

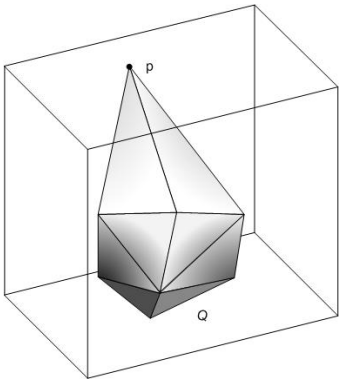

Figure 3: The hull after point p and the new faces have been added.

3 Implementation of Convex Hull Algorithm

To simplify the data structures used, the algorithm assumes that every face of the surface of the *polytope* (convex polyhedron) is a triangle. Three primary data types are required to define a convex hull, vertices, edges, and faces. The computer program that implements the algorithm consists of four basic sections: read, create initial polytope, construct the hull, and print. First the vertices are read into an array. Next, an initial polytope for the incremental algorithm is created. It is a double-sided triangle, a polyhedron with three vertices and two faces that are identical except for the order of their vertices. To construct this figure, three noncollinear points are found, and after marking them as processed the double-sided triangle is built as the first hull Q. It is now necessary to find a fourth nonplanar point p to form a tetrahedron. For each point p it must be determined if p is inside or outside Q. To accomplish this, one must determine for every face f of Q if the face f is *visible* from point p. The face f is visible from p iff it lies in the positive halfspace determined by the plane containing f. The positive side is determined by the counterclockwise orientation of f. If no face is visible from p, then it must lie inside the hull, and it is marked for deletion. If p is outside Q, then the hull is transformed by finding the tangent planes that bound a cone of triangle faces with the apex at point p and the base edges of Q (Figure 2). The portion of the polytope visible from p forms a connected region of the surface of the existing hull as indicated by the striped faces in Figure 2. The interior faces of the region must now be deleted and the cone connected to the boundary. To determine the edges of the hull, those edges adjacent to two visible faces are marked interior and marked for deletion; those edges with one adjacent visible face are identified as on the border of the visible region. Once the edges of the border are identified, then for every edge on the border a new triangular face can be constructed using that edge and the point p. At this point, the convex hull is almost complete. All that remains is to "clean up" the hull by deleting hidden faces and edges and making sure the vertex, edge and face lists are properly linked.

4 Construction of the Prototype

Our effort to model the algorithm began by building a visualization program using OpenGL. The program we devised first reads in a data set consisting of (x,y,z,r,g,b) giving position and color information and then creates a voxel for each point in an n x n x n space. The voxels are cubes that can be opaque or transparent. Once the scene has been rendered, it can be rotated clockwise or counterclockwise in the x, y, and z directions around a fixed point. Given a set of three-dimensional non-planar coordinates (a subset of the original data set) the program determines the convex hull defined by those points. In our program we identified the hull by changing the color of the points that constituted it. When operational the program demonstrated that the convex hull algorithm could be implemented in a graphics program.

Wishing additionally to implement the model into a three-dimensional touch environment, we developed a related program using the GHOST® API to incorporate haptics. The program we developed for use on the PHANTOM™ haptics device uses "voxel world" coordinates, where the position of the voxel is (x, y, z) and x, y, and z are all integers. As a trial experiment, we created a 3 x 3 x 3 grid of spheres to represent the voxel space. The spheres were large enough to be felt, but separated by sufficient space to permit movement of the haptic cursor within the grid. We made use of the PHANTOM™ haptics device as a three-dimensional selection device. Using the stylus, we selected certain spheres that represented the set of points to define our convex hull. The algorithm then determined the interior points of the hull and we identified those voxels by changing their color.

5 Future Work

One application of a tool such as we devised is to facilitate the exploration of three-dimensional scientific or abstract data sets using haptics. At the present we hope to incorporate this work into a tool kit that uses haptic feedback to explore Light Detection and Ranging (LIDAR) data. The LIDAR tool kit will enable the user to be immersed in and interact with a point cloud of data and the 3-D lasso editing tool will allow for selection of points for removal or coloring.

References

[1] Van Scoy, F., Peredera, A., and Kime, A. (1999), "A 3-D Lasso Tool for Editing 3-D Objects: Preliminary Work", Workshop on Virtual Reality, Marilia, Brazil, November 18-20.

[2] O'Rourke, J. (1998), *Computational Geometry in C*, second edition, Cambridge University Press, Cambridge.

[3] Graham, R.L. (1972), "An efficient algorithm for determining the convex hull of a finite planar set", *Inform. Process. Lett.* **1**, 132-3.

An Experimental Study of Performance and Presence in Heterogeneous Haptic Collaboration: Some Preliminary Findings

Margaret McLaughlin, Gaurav Sukhatme, Wei Peng, Weirong Zhu, and Jacob Parks
Integrated Media Systems Center, University of Southern California

Abstract

Based on a distributed architecture for real-time collection and broadcast of haptic information to multiple participants, heterogeneous haptic devices (the PHANToM and the CyberGrasp) were used in an experiment to test the performance accuracy and sense of presence of participants engaged in a task involving mutual touch. In the experiment, the hands of CyberGrasp users were modeled for the computer to which the PHANToM was connected. PHANToM users were requested to touch the virtual hands of CyberGrasp users to transmit randomly ordered letters of the alphabet using a pre-set coding system. Performance accuracy achieved by a small sample of participants was less than optimal in the strict sense: accurate detection of intended location and frequency of touch combined ranged from .27 to .42. However, participants accurately detected the intended location of touch in 92% of the cases. Accuracy may be positively related to pairwise sense of co-presence and negatively related to mean force, force variability, and task completion time.

1. Introduction

In many applications of haptics it will be necessary for users to interact with each other as well as with other objects. We have been working to develop an architecture for haptic collaboration among distributed users. Our focus is on collaboration over a non-dedicated channel (such as an Internet connection) where users experience stochastic, unbounded communication delays [1-5].

In our most recent work, we have focused on improving our system by implementing the synchronization mechanism as a collaboration library. To achieve the desired visual and haptic capabilities for the collaboration program, a model of a human hand was developed for the computer to which the PHANToM was connected. Each segment of the hand is a 3D model. Each component of the hand (the palm, the three segments of each finger, and the two segments of the thumb) was imported separately into the haptic environment. The components were arranged and aligned visually to produce the image of the hand. In the code, the components were organized in a tree-like architecture. To maintain the shape of the hand as it moves during simulation, the movements of components more distal to the wrist are described in the reference frames of adjacent components more proximal to the wrist. For example, the tip of a finger moves with respect to the middle segment of the finger, the middle segment moves with respect to the first segment of the finger, and the first segment moves with respect to the palm. This utilization of relative motion reduces the amount of information required to describe the motion of the hand. Only the rotation angles of each component (relative to its 'parent' component) are needed. The x-y-z coordinates of each component relative to its parent are fixed, and therefore need not be updated. (The only rectangular coordinates that update are for the wrist.) The CyberGrasp computer sends updated transform parameters, which the PHANToM computer uses to generate new coordinate values for the hand.

The other major issue concerning development of the hand model involved contact sensing by the PHANToM. In the haptic environment, regular, solid geometric shapes provide the PHANToM with substantial contact force. This was not so with the hand. Its two-dimensional triangular components would tend to let the PHANToM pass through, with little tactile indication of contact. This problem was solved in a makeshift fashion. A skeleton was constructed out of solid geometric shapes, and placed inside the hand model, to provide contact sensation for the PHANToM. In this solution, PHANToM users see the VRML hand, and feel the skeleton. (We will address this pass-through issue in future work by considering alternative surface representations.)

2. Experiment

2.1 Experiment Design

Here we report an experiment to evaluate our collaborative system. Some preliminary data were collected from a small-scale study of heterogeneous haptic collaboration, in which one participant, wearing the CyberGrasp, interacts over the Internet with another participant at a remote location (another lab area), who is assigned to a computer with the PHANToM haptic stylus attached. The participants are assigned to

an experimental task that assesses the ability of a subject wearing the CyberGrasp to recognize and discriminate among patterns of passive touch with respect to frequency and location, and the ability of the PHANToM user to communicate information effectively through active touch.

We can call the participants, respectively, P1 and P2. Participant P1's monitor shows a model of P2's (the CyberGrasp wearer's) hand, which is updated in real time to reflect the current location and orientation of P2's hand and fingers. P1's task is to use a PHANToM haptic stylus, the tip of which is represented by an enlarged round ball, to communicate information to P2 through a tactile "Morse code" which maps the number of times a particular digit is "touched" onto the 26 letters of the alphabet. A keypad showing this mapping is visible on P1's display. P2 holds her hand stationary during this process. P2 then enters (or has entered for her) the letters she thinks she has received into a keypad visible on her own display, from which they are logged into a word file. On P2's monitor, only the keypad is visible; P2 is unable to see the hand display. The two subjects can communicate with each other via a text-based messaging system, sending pre-programmed messages such as "Experiment begins," "New word," "New letter," "Ready for next letter," "Next trial," and so on. All text messaging between the partners is logged and time-stamped.

2.2 Participants

Participants were recruited through notices placed on bulletin boards and circulated through the campus BBS and student mailing lists. For participants to be eligible they had to be right-handed, over 18, and unacquainted with any other person planning to volunteer for the study. Upon acceptance into the participant pool, volunteers were randomly assigned to the "CyberGrasp" condition or the "PHANToM" condition and directed to one of two lab locations on campus. The target N of pairs for the study is 25; we are currently collecting data and report here on a very small sample of three pairs.

2.3 Procedure

Upon arrival at their respective lab locations, participants were then provided with a PowerPoint tutorial tailored to the type of haptic device to which they were assigned. The tutorial introduced them to haptics and in the case of the PHANToM user provided an opportunity for them to practice using the device for active exploration of a digital object. The CyberGrasp user's role in the experiment was to be a passive recipient of touch and as such did not "practice" but was simply taken through the calibration process upon completion of the tutorial.

The experimenters in their respective lab locations communicated with Motorola Talkabout radios to coordinate opening of the collaborative workspace. Participants joined in the workspace over an ordinary Internet connection. Next, they were provided with oral instructions which reinforced the explanations of the experimental task provided in the tutorials. As an added check to ensure that any obtained errors in location or frequency of touch were attributable only to the haptic interaction and not to participants' misreading of the code, participants were required to state aloud which finger they intended to touch and how many times they intended to touch it. When the oral instructions were completed, the investigator supervising the PHANToM user started a process for recording the haptic data stream (capturing and time-stamping applied force at 10ms intervals) [6] and signaled through the keypad that the experiment was to begin. The investigator supervising the CyberGrasp user minimized the collaborative workspace window so that the participant would respond only to the haptically communicated information. The pair of participants worked their way through a word list that contained all 26 letters of the alphabet, sorted randomly into nine sets of three- and two-letter words. At the conclusion of the trials, participants completed a post-task evaluation with items adapted from the Basdogan et al. measure of co-presence. [7] Among the questions of interest: Did the subject feel present in the haptic environment? Did the subject feel co-present with the remote partner? Did the subject believe that the remote partner was a real person? Subjects were also asked to make attributions about their partners with respect to standard dimensions of perception (unsocial-social; sensitive-insensitive; impersonal-personal; cold-warm) (Sallnas, Rassmus-Grohn, & Sjostrom, 2001). [8] (Findings with respect to the person perception data will be reported elsewhere.)

3. Results

3.1 Accuracy

Performance for all three pairs was seemingly poor. Accuracy of the best pair was 42.3%. The next best pair got 38.4% of the letters correct, and the least effective pair only 26.9%. However, examination of the confusions data indicates that, with the exception of the thumb (see Table 1, below), most of the confusions

resulted from an overestimate of the number of times the finger was tapped. Only 8% of the errors made were egregious, e.g., the recipient was confused about which finger was being tapped. We present two of the confusions matrices below, the matrix for the ring finger (Table 1) and for the thumb (Table 2).

Table 1. Table of Confusion for the ring finger *

	Reported Number of Taps						
		1	2	3	4	5	6
Number of Taps Called for by Code	1		x		x		
	2			x			
	3				xx x		
	4			xx			
	5						
	6						

*Confusions as to number of taps are represented by small x's.

Table 2. Table of Confusion for the thumb*

	Reported Number of Taps								
		1	2	3	4	5	6	Finger Mid.	 Pinky
Number of Taps Called for by Code	1		x						**X**
	2	x							
	3	x					x		
	4		x					**X**	
	5	xx					x		
	6					xx			

*Confusions as to number of taps are represented by small x's. Confusion as to location of touch are represented in bold capital X's at the right-hand side of the table

3.2 Performance and presence variables

Performance values for the participant pairs for (a) the number of points at which there was measurable force, (b) minimum force, (c) maximum force, (d) force variability (standard deviation of sampled values), (e) time to task completion, and (f) accuracy (percentage of correctly decoded letters) are presented in Table 3 (a)(b). Co-presence data is also reported in Table 3 (b). Comparison of the most accurate pair (Pair 3) and least accurate pair (Pair 1) indicates that the less accurate pair had far fewer sampled points with measurable force, greater mean force, higher force variability, and longer time to task completion (although taken over all pairs these relationships are not monotonic with respect to task completion time). Comparison of the most and least accurate pairs (Pair 3, Pair 1) indicated that perceived co-presence was rated higher by the more accurate subjects, and by both members of the pair. However, taken over all pairs the relationship between co-presence and accuracy is not monotonic for the PHANToM users.

Table 3. Performance variables and co-presence ratings for participant pairs

(a)

Pair	N Points Measurable Force	Min Force	Max Force	Mean Force	Force S.D.
1	0494	.0005	.8530	.2788	.2623
2	6607	.0005	.8974	.2278	.2501
3	3688	.0005	.9039	.1240	.1699

(b)

Pair	N Points Measurable Force	Min Force	Max Force	Mean Force	Force S.D.
1	0494	.0005	.8530	.2788	.2623
2	6607	.0005	.8974	.2278	.2501
3	3688	.0005	.9039	.1240	.1699

4. Discussion

Although the accuracy levels obtained in this small sample appeared to be low, it was in fact the case that in 92% of the instances the passive touch recipient (the CyberGrasp wearer) was able to distinguish which digit was being touched. The bulk of the obtained errors resulted from an overestimate by one of the number of times the digit was tapped. There are a number of possible explanations, two of which seem most likely. First, there may have been incidental vibration of the CyberGrasp unrelated to the intended activity of the PHANToM user. If that were the case, however, there would have been more reports by the

CyberGrasp user that his or her partner was "trying to touch several fingers" at once. Our experience to date indicates that touch on some of the middle digits can produce vibration in neighboring digits. What we did observe is that the PHANToM user, having only a single point of contact, often "missed" connecting with the partner's fingers on the first several tries at a new letter, and would frequently follow up on a light application of force to the finger with a stronger one, even though the deformation of the finger model on the first occasion of contact was clearly visible on the monitor. Further practice and more intensive coaching in the tutorial on interpreting the visual indicators of contact should improve results in future experiments. Incorporation of immediate feedback about performance accuracy on initial trials should also improve performance on subsequent trials.

Poorest performance was obtained for information communicated through taps to the thumb. In addition to the two cases in which the CyberGrasp wearer perceived a tap intended for the thumb to be directed to another digit, the thumb confusions matrix shows clearly that the passive recipient consistently underestimated the number of taps relative to the number intended by the active partner. The most plausible explanation has to do with the comparative difficulty given the orientation of the hand of contacting the palmar face of the thumb with the PHANToM. Many of the taps were applied to the dorsal side.

Although we expected that greater mean force would be associated with greater accuracy; that does not appear to have been the case in this small sample. There may be a force threshold which once met is adequate for detection, or it may be that the necessary level of force varies with receivers. And it may be that mean force over multiple trials is not a meaningful measure. We hope to sort out these issues as we collect additional data. Our expectations with respect to number of points of measurable force, force variability and task completion time are so far being confirmed with the data at hand; more variable application of force, less frequent application of force, and longer time to complete the task appear to have a negative impact on pairwise accuracy.

Finally, there is some very slight evidence that sense of co-presence may turn out to be stronger in more accurate pairs, although we are far more comfortable looking for trends in performance over multiple trials with a handful of participants than we are in looking for relationships between overall performance accuracy and subjective, retrospective assessments of the collaborative environment given only the three pairs. We expect to find additional evidence in support of this relationship as we continue to collect data.

References:

[1] Hespanha, J., McLaughlin, M. L., & Sukhatme, G. (2002). Haptic collaboration over the Internet. In McLaughlin, M. L., Hespanha, J., & Sukhatme, G. (Eds.). *Touch in Virtual Environments*: *Haptics and the Design of Interactive Systems*. Prentice Hall.

[2] Hespanha, J., Sukhatme, G., McLaughlin, M., Akbarian, M., Garg, R., & Zhu, W. (2000). Hetereogeneous haptic collaboration over the Internet. *Proceedings of the Fifth Phantom Users' Group Workshop*, Aspen, CO.

[3] McLaughlin, M. L., Sukhatme, G., Hespanha, J., Shahabi, C., Ortega, A., & Medioni, G. (2000). The haptic museum. *Proceedings of the EVA 2000 Conference on Electronic Imaging and the Visual Arts*. Florence, Italy.

[4] Shahabi, C., Ghoting, A., Kaghazian, L., McLaughlin, M., & Shanbhag, G. (2001). Analysis of haptic data for sign language recognition. *Proc. First International Conference on Universal Access in Human-Computer Interaction (UAHCI),* New Orleans, Louisiana, 5-10 August 2001. Hillsdale, NJ: Lawrence Erlbaum.

[5] Sukhatme, G., Hespanha, J., McLaughlin, M., & Shahabi, C. (2000). Touch in immersive environments. *Proceedings of the EVA 2000 Conference on Electronic Imaging and the Visual Arts*, Edinburgh, Scotland.

[6] Shahabi, C., Kolahdouzan, M., Barish, G., Zimmermann, R., Yao, D., & Fu, L. (2001, June). Alternative techniques for the efficient acquisition of haptic data, ACM SIGMETRICS/Performance 2001, Cambridge, MA.

[7] Basdogan, C., Ho, C., Srinivasan, M. A., & Slater, M. (2000) An experimental study on the role of touch in shared virtual environments. ***ACM Transactions on Computer Human Interaction***, ***7***(4), 443-460.

[8] Sallnas, E. L., Rassmus-Grohn, K. & Sjostrom, C. (2001). Supporting presence in collaborative environments by haptic force feedback. ***ACM Transactions on Computer-Human Interaction***, ***7*** (4), 461-476.

Assessing the increase in haptic load when using a dual PHANToM setup

Joan De Boeck, Chris Raymaekers, Karin Coninx

Expertise Centre for Digital Media, Limburg University Centre,
Wetenschapspark 2, B-3590 Diepenbeek, Belgium
{joan.deboeck, chris.raymaekers, karin.coninx}@luc.ac.be

Abstract

Two handed input and collaboration are currently two active research areas in the domain of virtual environments. In a haptic application in particular, by means of a PHANToM device, two possibilities to build such a setup exist: either by integrating two PHANToM interface cards in one computer, or by using two networked computers, each equipped with one PHANToM interface card. The former method is likely to increase the computer's haptic load, while the latter adds extra code complexity and latency in the interaction, but does not augment the haptic load. Since very little can be found about the consequences of both solutions, this paper will describe an experimen, that measures the increase in haptic load of a dual PHANToM setup over a standard configuration.

1 Introduction and related work

Although the use of force-feedback is a huge improvement in the interaction with virtual environments, current implementations are mostly restricted to a single point in space. To achieve a second interaction point, alternatively, a second interface card can be installed in order to drive a second PHANToM device, but this is very likely to increase the computer's haptic load. On the other hand, when using a collaborative application, multiple computers, distributed across the network, are all connected to their own haptic device [ALHAL01][HESPA00]. However, this setup adds extra code complexity and latency in the interaction. As an example, our research into the virtual percussionist application [DEBOE02] adopts those principles of distributed collaborative setups and applies them in a two handed input application. We believe this extra coding complexity might be justified when the increase in the haptic load gets too high in the single computer solution. At this moment however, no formal research has been conducted into this increase in a dual PHANToM setup. Some experiments on the computational load in a single PHANToM setup have been performed: Acosta et al [ACOST02] measured the maximum complexity in terms of number of objects and object complexity across different GHOST versions. Anderson et al [ANDER02] compared the performance of the GHOST API with e-Touch API for large complex objects. This paper, as a part of our research in two-handed haptic input, extents the above-mentioned research. The next sections will describe an experiment, which compares the haptic load of a dual PHANToM setup and a single PHANToM setup and discusses on the results.

2 Experimental Setup

This paper elaborates on two experiments. In a first experiment, we have measured the haptic load in a scene that contains one single object with increasing complexity. This object has been created by subdividing either a tetrahedron or a cube using the Loop subdivision scheme [LOOP87][RAYMA01]. Each subdivision level is represented in the haptic scene graph, by an instance of gstTriPolymeshhaptic. In order to test if the haptic load depends on the object's geometry, we also have used more natural models like a rabbit and a fish.
In a second test, we measured the haptic load in a scene with an increasing number of objects. These objects are positioned in a 3D matrix (as in [ACOST02]) which can grow in each direction. The objects do not intersect each other's bounding box and they also varied in complexity, using the same techniques as in the first experiment.
Both experiments have been conducted with a single and a dual PHANToM setup. The criterium measured in the two experiments is the haptic load, as measured with Sensable's Haptic Load (HLOAD) tool.

The computer used in our experiments was a Pentium III 600 MHz, 256 MB RAM, running Windows NT SP6. However, due to an incompatibility between the PHANToM PCI interface card, the AGP adapter and the computer's motherboard, we were forced to conduct our tests with a poor video card. Since the GHOST thread is a high priority thread, we believe this has little or no effect to the haptic load measured.
Our tests have been conducted on Windows NT, because we did not succeed in connecting two PHANToM devices on Windows 2000[1]. Our assumption that the choice of our hardware and OS does not influence our results are confirmed by comparing our single PHANToM test results on a Pentium III 850 MHz and a dual Pentium III 800 MHz running Windows 2000.

3 Results

3.1 Experiment 1: Objects with increasing complexity

			1 Phantom			2 Phantoms		
Subdivision		**#tri**	**N Contact *1P/0C***	**Contact *1P/1C***	**Contact & Moving**	**N contact *2P/0C***	**1 Contact *2P/1C***	**2 Contacts *2P/2C***
Tetraedron	level 0	4	<10	>10		10	10< - <20	<20
Cube	level 0	12	<10	>10		>10	10< - <20	<20
Tetraedron	level 1	16	<10	>10		>10	10< - <20	<20
Cube	level 1	48	<10	>10		>10	10< - <20	<20
Tetraedron	level 2	64	<10	>10		>10	10< - <20	<20
Cube	level 2	192	<10	>10		>10	10< - <20	20
Tetraedron	level 3	256	<10	>10		>10	10< - <20	20
Cube	level 3	768	<10	>10		>10	10< - <20	>20
Tetraedron	level 4	1024	<10	>10		>10	10< - <20	>20
Cube	level 4	3072	<10	>10		>10	10< - <20	>20
Tetraedron	level 5	4096	<10	>10		>10	10< - <20	>20
Cube	level 5	12288	<10	>10		>10	>20	>30
Tetraedron	level 6	16384	<10	<20		>10	>20	>30
Cube	level 6	49152	<10	20	30	>20	40	<60
Tetraedron	level 7	65536	<10	30	40	<20	40	>70
bunny		69451	<10	50	60	<20	70	quit
Fish		100480	<10	30	90	20	Unstable	quit
rabbit		134074	<10	80	quit	<20	quit	quit
Thetraedron	level8	262144	<10	70	quit	quit	quit	quit

Table 1: haptic load in a single and double PHANToM Setup with complex objects.

Table 1 summarizes the results of the first experiment. The values indicate a percentage of the haptic load of the simulation. When looking at the first column where no contact is made, one can see a quite constant haptic load both in the single PHANToM (1P/0C) as in the dual PHANToM setup (2P/0C). When touching the object (without moving the surface contact point over the surface) (1P/1C) the haptic load starts increasing from a shape with 160,000 triangles. This is consistent with the values reported in [ANDER01]. As can be seen in the next column, the haptic load augments when the surface contact point moves over the object's surface. This causes the GHOST-thread to quit with the most complex models in our experiment.
With the dual PHANToM condition, the haptic load again is quite stable when no contact is available (2P/0C), but increases slightly with increasing the object's complexity. The haptic load when one PHANToM touches the object (2P/1C) is somewhat higher, compared with the single PHANoM setup. The table here indicates a higher increase for the more complex models. The second surface contact points introduces another increase in such that the total haptic load is roughly 50% more than in (1P/1C). The results of those experiments are graphically depicted in fig 1.

[1] Another research lab reported us the same technical problems when running Windows 2000.

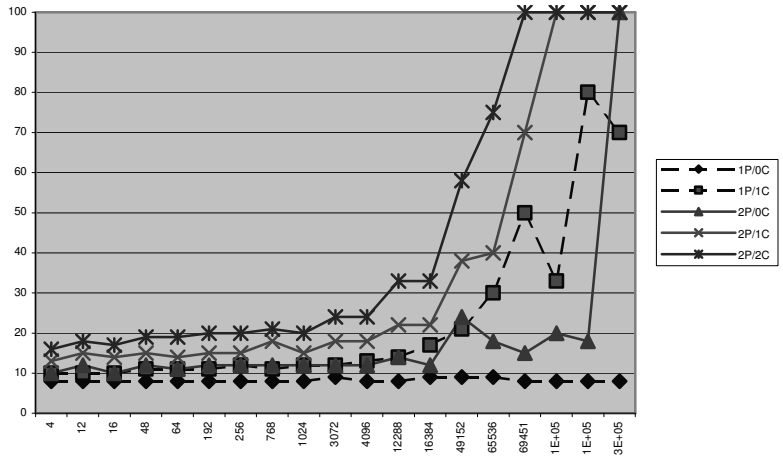

Fig. 1. Graph of the haptic load in a single and double PHANToM Setup with complex objects.

3.2 Experiment 2: Scenes with increasing complexity

Tetrahedron Subdivision Level 0 (4 triangles)					
	One PHANToM		Two PHANToMs		
# objects	**No Contact**	**1 Contact**	**No Contact**	**1 Contact**	**2 Contacts**
8	10	<20	>20	>30	>40
27	<20	>30	30	<50	>60
64	>30	<70	<60	quit	quit
125	<50	quit	quit	quit	quit
216	quit	quit	quit	quit	quit

Table 2: haptic load with multiple objects in the scene. (subdivision level 0)

Tetrahedron Subdivision Level 2 (64 triangles)					
	One PHANToM		Two PHANToMs		
# objects	**No Contact**	**1 Contact**	**No Contact**	**1 Contact**	**2 Contacts**
8	>10	20	<20	>20	30< - <40
27	<20	30< - <40	>30	>50	<70
64	>30	50< - <60	<70	quit	quit
125	<70	quit	quit	quit	quit
216	quit	quit	quit	quit	quit

Table 3: haptic load with multiple objects in the scene. (subdivision level 2)

Tetrahedron Subdivision Level 4 (1024 triangles)					
	One PHANToM		Two PHANToMs		
# objects	**No Contact**	**1 Contact**	**No Contact**	**1 Contact**	**2 Contacts**
8	10	<20	70	quit	quit
27	20	>40	quit	quit	quit
64	<40	60	quit	quit	quit
125	<70	quit	quit	quit	quit
216	quit	quit	quit	quit	quit

Table 4: haptic load with multiple objects in the scene. (subdivision level 4)

Tables 2, 3 and 4 show the results of the haptic load in a scene with an increasing number of objects. Each table displays the same number of objects, but with more triangles per object. All the objects are placed in a grid in such a manner that their bounding boxes do not overlap. A single PHANToM setup can fully support a scene with up to 64 tetrahedrons, while the dual PHANToM setup has the same load when simulating only 27 objects. The results of table 4 show that the haptic loop with 2 PHANToMs quits when touching one of the level 4 subdivision-objects in the scene, while the single PHANToM setup still supports 64 objects.

4 Discussion

4.1 Results

Our values in the single PHANToM case correspond to the findings of [ACOST02] and [ANDERS02]. Although, [ACOST02] can support up to 600 cubes, while our simulation already quits at 125 objects. We suppose this is caused by the standard *gstCube* (used in [ACOST02]), which is simpler and more efficient than the *gstTriPolymesh* used in our experiment.
From the first experiment, we can conclude that the haptic load in a dual PHANToM setup has been increased, compared to the single PHANToM setup. As long as the object is relatively simple (up to 16,000 triangles) the haptic load keeps below 30%, which results in a stable simulation.
Although the HLOAD application, which was the only tool we had to measure the haptic load, is not the most accurate tool one can imagine, we roughly can say that the second PHANToM increases the haptic load with about 50%. This can be confirmed by comparing the number of triangles in the most complex, but stable simulation in the first condition (fish with 100,480 triangles) with the maximum number of triangles in a stable dual PHANToM simulation (tetrahedron level 7 with 65,536 triangles).
Even more pronounced is the increase of the haptic load in a scene with multiple objects. If we compare "single contact with one PHANToM" (1P/1C) with a "double contact with two PHANToMs" (2P/2C), we see that the haptic load almost doubles. This makes that complex scenes, which can be run with one PHANToM, are not supported by a dual PHANToM setup.

4.2 Other findings

During the course of our experiments, we have encountered a number of interesting situations. Some of these appear to be obvious, but we believe they can give the programmer a better understanding of optimizing a more complex scene.

- When starting the haptic loop, very often a "haptic load spike" is encountered. This is why heavy models often crash at the beginning of the simulation. However if a complex simulation accidentally "survives" a startup, the simulation seldom is completely stable. Most of the time the haptic thread quits when acting on the scene.
- When exploring our "natural" objects (fish, rabbit), the haptic load was higher when approaching a more complex region with lots of small triangles.
- When sliding the PHANToM over an object, this increases the haptic load, compared to a static contact. On the other hand a static contact, which touches more than one triangle (e.g. in a corner), requires more processing time.
- We could not make Windows 2000 to work with the two PHANToMs, although we have conducted some of our single PHANToM tests on two Windows 2000 PCs, as well. In general we can state that there is little difference between the results of these tests and the Windows NT test.
- At first sight, the dual processor seems to perform better in starting-up a very complex scene: scenes that quit at start-up, do run on the dual processor computer. When interacting in those scenes, however, the results are quite similar to a single processor machine.
- On a dual processor computer other tasks will slow down less when executing a heavy GHOST thread. For instance, when running complex scenes, the haptic loop will slow down the graphics on a single processor computer, which is not true on a dual processor machine. This is quite obvious because the ghost thread in some cases can take up to about 99% of one processor's time, which is only 50% of the total processing power of a dual machine.

5 Conclusions and future work

For a multiple-contact interaction with a PHANToM device, two possibilities exist: most commonly, a second PHANToM will be attached to the same computer, or alternatively two computers in a network, both connected to a separate PHANToM share the same virtual scene. The first solution increases the computational load; the next solution introduces network delays. As a step to a well-grounded choice between those two options, in this paper we have conducted a test to measure the increase of the haptic load of a common dual PHANToM setup. Because we could not make a dual setup to work under Windows 2000, we have conducted our experiment under Windows NT SP6.

The experiment consisted of two tests: one that measured the haptic load in respect to the complexity of one object, and a second experiment, which tested the haptic load in respect to the scene complexity.
We can conclude that the increase of the haptic load is quite significant (roughly about 50% in experiment 1 and about 100% in experiment 2), but this is of less importance when running small scenes (1 object of less than 16,000 triangles, or a scene of less than 30 objects). We want to conclude that for large or complex scenes the dual PHANToM setup clearly has its limitations. In such a case, one can consider to afford the extra code complexity for a distributed setup, although network delay certainly will be a constraint in this case.

6 Acknowledgements

We first of all want to thank Geert Van De Boer for his coding effort in realizing this test and getting a dual PHANToM setup to work. Next, we also want to thank Jan van Erp en Hendrik-Jan van Veen from TNO (the Netherlands) for their suggestion to install Windows NT instead of Windows 2000.

7 References

[ACOST02] Scene Complexity: A measure for real-time stable haptic applications, E.Acosta, B.Temkin, October 2001, Aspen; CO, USA. 2001

[ALHAL01] Tele-Handshake: A Cooperative Shared Haptic Virtual Environment, M.O.Alhalabi, S.Horiguchi, EuroHaptics 2001, July 2001, Birmingham, UK.

[ANDER02] The ActivePolygon Polygonal Algorithm for Haptic Force Generation, T. Anderson, N. Brown, Novint Technologies, PUG2001, October 2001, Aspen; CO, USA. 2001

[DEBOE02] A networked two-handed haptic experience: The Virtual Percussionist, J. De Boeck, P. Vandoren, K. Coninx, VRIC2001, June 2002, Laval, France, pp 131-139

[HESPA00] Haptic Collaboration over the Internet, J. Hespana, M. McLaughlin, G. Sukhatme, M. Akbarian, R. Grag, W. Zhu, Proceedings of the fifth PHANToM Users Group Workshop; October 28-30, Aspen; CO, USA. 2000

[LOOP87] Smooth Subdivision Surfaces Based on Triangles, C. Loop, University of Utah, Department of Mathematics, Utah, US, August 1987

[RAYMA01] Fast Haptic Rendering of Complex Objects Using Subdivision Surfaces, C. Raymaekers, K. Beets; F. Van Reeth, PUG2001, October 2001, Aspen; CO, USA. 2001

Comparison of Human Haptic Size Discrimination Performance in Real and Simulated Environments

Marcia Kilchenman O'Malley
Mechanical Engineering and Materials Science
Rice University
Houston, Texas
omalleym@rice.edu

Abstract

The performance levels of human subjects in size discrimination experiments in both real and virtual environments are presented. The virtual environments are displayed with a Phantom desktop three degree-of-freedom haptic interface. Results indicate that performance of the size discrimination tasks in the virtual environment is comparable to that in the real environment, implying that the haptic device does a good job of simulating reality for these tasks. Additionally, performance in the virtual environment was measured at below maximum machine performance levels for two machine parameters. The tabulated scores for the perception tasks in a sub-optimal virtual environment were found to be comparable to that in the real environment, supporting previous claims that haptic interface hardware may be able to convey, for these perceptual tasks, sufficient perceptual information to the user with relatively low levels of machine quality in terms of the following parameters: maximum endpoint force and maximum virtual surface stiffness. Results are comparable to those found for similar experiments conducted with other haptic interface hardware, further supporting this claim.

1 Introduction

This paper presents a comparison of human haptic performance in real and virtual environments. Results support the case that haptic interfaces are good at simulating real objects, and indicate that they can do so without excessive machine performance demands for the tasks described here. Comparisons of performance in real and virtual environments have been made in the past. Typically these comparisons are made with the virtual environment display operating such that the best achievable representation of reality is presented to the user. For example, completion times for a pick and place task performed in a real-world control environment and in three virtual conditions were presented by Richard et al. [1]. Their findings showed that for the pick and place task, completion times, a measure of task performance, were lower for the real-world control environment than for each of the three virtual environments tested. However, accuracy for depth and lateral placement were comparable for one haptic display and the real-world control. Similarly, Buttolo et al. [2] used comparative methods to study the differences in performance of simple manipulation tasks with real objects, with a virtual reality simulation containing force feedback, and remotely with a master and slave system, also with force feedback. Their findings also showed that performance with the virtual environment was similar to that with real objects. This paper takes a similar comparative approach to verify the quality of a haptic device in simulating realistic virtual environments for a simple perceptual task of size discrimination, where subjects determine which of two objects placed side by side is larger. Additionally, the quality of the haptic device, in terms of two parameters, is degraded and another performance comparison is made.

2 Methods

2.1 Virtual Environment Apparatus

The Phantom desktop was used to simulate the virtual environments. Hardware specifications are listed in Table 1.

Table 1. Phantom Desktop hardware specifications

Workspace	16x13x13 cm
Maximum force	6.4 N
Maximum continuous force	1.7 N
Force feedback	3 DOF
Position sensing	6 DOF

2.2 Testing Environments

In both the real and virtual environments, subjects held a stylus with the dominant hand and entered responses on a computer keyboard with the non-dominant hand. The dominant hand was shielded from view with a curtain for both tests. A frame for the curtain was constructed from a cardboard box. The face closest to the subject was cut out and replaced with a curtain to

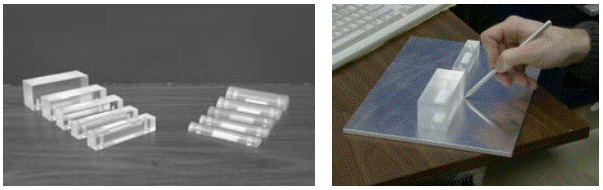

Figure 1. Photograph of the real blocks and the environment for a square ridge size discrimination task.

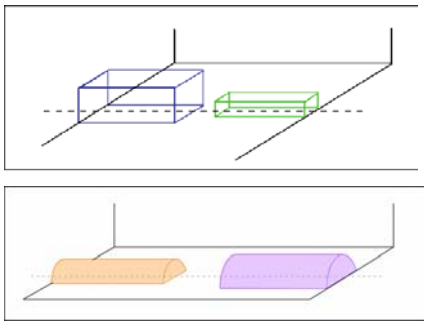

Figure 2. 3-D model of the simulated environment for the square and round ridge size discrimination tasks

Figure 3. Test subject seated at testing station for virtual environment experiments. (The box and curtain used to obstruct the subject's view are removed in this picture)

allow free movement of the hand. The back of the box was also removed to allow the test administrator to change blocks for the real environment. The box was secured to the desktop throughout the practice and testing sessions.

2.2.1 Real Environment

The real environment consisted of square and round cross-section blocks, machined out of acrylic, mounted on pegs on an aluminum plate. The blocks were covered with clear contact paper to account for any texture irregularities due to machining. The subject used an aluminum stylus to explore the environment. These blocks and the test plate are shown in Figure 1.

2.2.2 Virtual Environment

Square and round cross-section blocks were displayed haptically with the Phantom Desktop. Figure 2 shows a visual representation of the virtual environment. Subjects explored the environment with the Phantom stylus as shown in Figure 3.

2.3 Experimental Paradigms

Perception experiments were conducted for ridges of square and semicircular cross-sections, and were conducted in a real environment and several virtual environments (high fidelity, low fidelity-force, and low fidelity-stiffness). The order of testing was varied for each subject to ensure that learning effects were not a factor. Short practice sessions (10 trials each) were conducted prior to each experiment.

2.4 Subjects

Ten test subjects were used for the size discrimination experiments. A cross-section of subject types (gender, dominant handedness, and experience with haptic devices) was chosen for each of these experiments.

2.5 Procedures

In each experiment, the subject was asked to feel the exterior of the two ridges and determine which was larger, entering their response on the keyboard (left or right). One of the two ridges was always the base size, with an edge length of 20 mm. The second ridge had an edge length of 20, 22.5, 25, or 30 mm. In both the real and virtual environment experiments, twenty trials of each stimulus pair were presented to the subject. In all, subjects sat for eight test sessions of eighty trials which were each preceded by a ten trial practice session.

2.5.1 Machine Parameters

In order to create low-fidelity environments, two machine parameters were selected to describe haptic interface machine performance, namely maximum force output and time delay. Force output correlates to torque output limits of motors, and increased torque output requirements are typically proportional to motor cost and size. Time delays are unfavorable in a real-time system, and reduction of time delay usually requires faster computing speed and higher quality electronics, each coupled to an increase in price. These two quantifiable machine parameters are easily understood by designers and are typical measures of system quality. During experimentation in the low-fidelity virtual environments, these machine parameters were lowered to minimum levels for good performance of the size discrimination task as determined by O'Malley and Goldfarb [3, 4]. To limit the force output of the manipulator, the output command force was saturated at 4 N for each trial [3]. This was accomplished by creating new classes in GhostSDK called WeakCube and WeakCylinder that take the maximum output force as an input. These classes were

based upon the GstCube and GstCylinder classes in GhostSDK. For the stiffness low fidelity simulations, *k* was set to 450 N/m and *b* to 45 Ns/m as recommended in [4].

2.5.2 Experiments

Experiment 1 – Real

Testing was conducted with the acrylic blocks and aluminum stylus for both square (A) and round (B) cross-section ridges.

Experiment 2 – High-Fidelity Virtual

Testing was conducted with the Phantom Desktop at default values for force and stiffness. Tests were conducted for both square (A) and round (B) cross-section ridges.

Experiment 3 – Low-Fidelity Virtual: Force

Testing was conducted with the Phantom Desktop at the default value for stiffness and a maximum output force of 4 N. Tests were conducted for both square (A) and round (B) cross-section ridges.

Experiment 4 – Low-Fidelity Virtual: Stiffness

Testing was conducted with the Phantom Desktop at the default value for force and a virtual surface stiffness of 470 N/m. Tests were conducted for both square (A) and round (B) cross-section ridges.

3 Results

Results for all experiments are presented in Figures 4 (square cross-section ridges, all environments) and 5 (round cross-section ridges, all environments). Results are shown as percent correct scores and are the average results across all test subjects. Standard errors are shown with error bars.

4 Discussion

In Figure 4, we see that performance in the high fidelity virtual environment is comparable to that in both low-fidelity virtual environments for size discrimination of ridges with square cross-section. At a size difference of 1.25 mm, performance is best in the real environment, although this result is not significant as determined by an analysis of variance (ANOVA). At all other sizes, performance in the real environment is comparable to that in the high and low fidelity environments.

Figure 5 shows results for all round cross-section size discrimination experiments. Again, we see comparable performance between the high and low fidelity virtual environments at all size differences. At the 1.25 and 2.5 mm size differences, performance appears to be slightly better in the real environment,

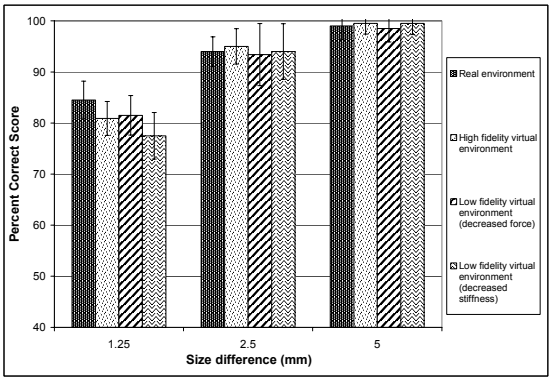

Figure 4. Results for all size discrimination experiments with square cross-section ridges (1A, 2A, 3A, and 4A)

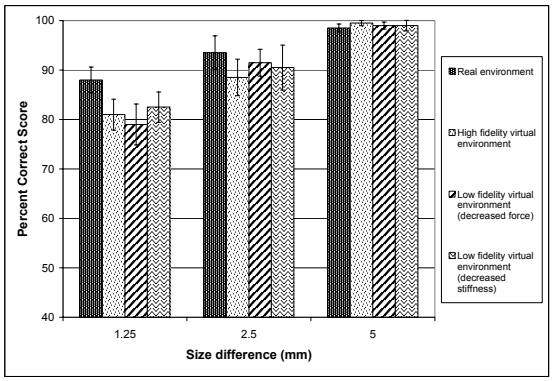

Figure 5. Results for all size discrimination experiments with round cross-section ridges (1B, 2B, 3B, and 4B)

although this result is not significant by the ANOVA. Overall, the two-way ANOVAs showed that there were no significant differences in performance when comparing based on environment (real, high fidelity virtual, low fidelity virtual-force, or low fidelity virtual-stiffness).

One subject commented that the low fidelity didn't feel much different than the high-fidelity. This comment supports the author's claim that low fidelity environments may be sufficient for some perceptual tasks.

5 Conclusions

The findings of these experiments, in which performance in a real environment was compared to performance in an environment displayed with a Phantom desktop for a size discrimination perception task, indicate that the Phantom does a fairly good job of approximating reality for the block environments described here. Not only do these results support the case that haptic interfaces are good at simulating real environments for these perceptual tasks, but they also

show that they can do so without excessive machine performance demands. Results are similar to those found for other haptic interface hardware.

6 Acknowledgements

The work presented herein was supported by NASA Grant NGT-8-52859. The author wishes to thank the volunteers who served as test subjects and the following students who contributed to this work: Andrea Lubawy, Matt Cuddihy, Tony Kellems, and Shannon Hughes.

7 References

[1] P. Richard and Ph. Coiffet, "Dextrous Haptic Interaction in Virtual Environmnets: Human Performance Evaluations," Proceedings of the IEEE International Workshop on Robot and Human Interaction, pp. 315-320, 1999.

[2] P. Buttolo, D. Kung, and B. Hannaford, "Manipulation in Real, Virtual, and Remote Environments." Intelligent Systems for the 21st Century. Systems, Man, and Cybernetics, Vol. 5, pp. 4656-4661, 1995.

[3] M. O'Malley and M. Goldfarb, "The Effect of Force Saturation on the Haptic Perception of Detail." IEEE/ASME Transactions on Mechatronics, Vol. 7(3), pp. 280-288, 2002.

[4] M. O'Malley and M. Goldfarb, "The Implications of Surface Stiffness for Size Identification and Perceived Surface Hardness in Haptic Interfaces," Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1255-1260, 2002

EVALUATION OF THE PHANToM PLAYBACK CAPABILITY

Robert L. Williams II and **Mayank Srivastava**
Department of Mechanical Engineering
Ohio University

Robert R. Conatser Jr. and **John N. Howell**
Department of Biomedical Sciences
Ohio University

ABSTRACT

This article presents implementation and evaluation of a position and force haptic playback system for the PHANToM haptic interface, in the context of our Virtual Haptic Back Project at Ohio University. Playback has the potential to improve virtual palpatory diagnosis training by allowing students to follow and feel an expert's motions prior to performing their own palpatory tasks. No human factors data is presented; rather, this article studies the performance and implementation of our playback system, in terms of how faithful the reproduction of recorded position and force is. We experimentally study the position and force errors upon playback, as a function of our playback parameters: spring stiffness *k* and zero force radius *r*. Position error decreases with increased *k* and decreased *r*. However, one cannot increase *k* or decrease *r* indefinitely as an unacceptable buzzing effect arises. Force error is not much affected by different *k* and *r*.

1. INTRODUCTION

The Virtual Haptic Back is under development at Ohio University to augment the palpatory training of Osteopathic Medical Students and Physical Therapy and Massage Therapy students (Holland et al., 2002). This project has implemented a high-fidelity graphical and haptic model of the human back on a PC, using the PHANToM interface for haptic feedback.

The current article focuses on the implementation and performance of our PHANToM playback feature, motivated by training needs in the Virtual Haptic Back Project at Ohio University. This article presents position and force error data in PHANToM playback in the Virtual Haptic Back context. This article first presents a brief overview of the Virtual Haptic back, followed by a description of our PHANToM playback capability, and then presentation and discussion of our playback system experiments and results.

2. THE VIRTUAL HAPTIC BACK

A virtual back graphics model has been developed, based on measurements taken with a 3D digitizer. Haptic feedback has been programmed, associated with this virtual back model via the PHANToM haptic interface (Fig. 2, Massie and Salisbury, 1994, also http://www.sensable.com/ .

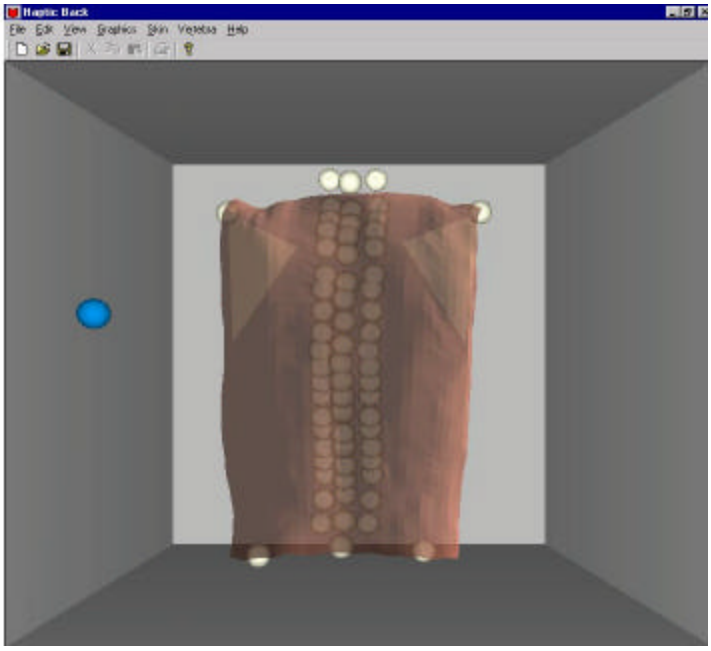

FIGURE 1. THE VIRTUAL HAPTIC BACK

The virtual back consists of skin, a spine (C2 at top then C6 through L5), interspinous ligament, scapulae, acromion processes and PSIS (posterior superior iliac spine). In using the Virtual Haptic Back, the student first encounters resistance from compression of the skin and then additional resistances representing underlying bone. The interspinous ligaments joining the spinous processes are palpated as objects with less intrinsic stiffness (more give) than the spinous processes. Transverse processes can also be palpated lateral to the spinous processes and deeper. Each vertebra can rotate in response to pressure applied by the operator to the transverse processes. The resistance to rotation can be set independently for each vertebra. The initial position of each vertebra can also be set independently via menu. The graphics can be set to reveal the underlying bone or not, so that the palpation can be done with or without the aid of seeing the vertebrae on the screen (the real world does not allow this choice!).

3. PLAYBACK SYSTEM

In a paper on diagnosing prostate cancer (Burdea et al., 1999), the PHANToM playback mode is used both to analyze a trainee's performance and to show the trainee how an expert approaches prostrate examinations. The same research group is applying general graphics playback in palpation training for detecting subsurface tumors (Dinsmore et al., 1997). In this paper, a data file is written with all inputs from all I/O devices to replay the user's actions graphically. This case does not involve the PHANToM with haptic playback. A second group is using the PHANToM playback feature in their horse ovary palpation simulator (Crossan et al., 2000), to implement a tutor/trainee model.

We have developed a haptic playback system wherein user's position and force interactions with a haptic model may be saved and played back to the PHANToM. This haptic playback has two versions, one in which the recorded interaction forces are played back and the haptic model is turned off and the other in which the recorded forces are not sent but the haptic model is turned on, i.e. we make the PHANToM trace the user's previous path and forces are felt due to the PHANToM interacting with the haptic objects.

To achieve playback in the Virtual Haptic Back, two data files are created during the recording mode. One file records the *XYZ* positions of the PHANToM and the other records its F_X, F_Y, and F_Z reaction forces. In the original simulation the input comes from the user's hand motions, via the PHANToM encoders. In playback the input is read from the data files; the position playback is achieved as described below.

Position Playback. For recording the user's path, points are sampled at a rate of 1000 *Hz*, and saved in a binary text file. These points are read and the PHANToM playback driving force *F* is calculated using (1). This driving force moves the tip of the PHANToM back through the previously-recorded positions for playback.

$$F = k\left(\|v\| - r\right)\frac{1}{\|v\|}v \qquad (1)$$

In (1), k is the virtual spring constant; v is the vector distance between the current PHANToM position and the next playback point to move to (the center of an attractive spherical force field, see Fig. 2). r is the radius of the spherical region in which there is no force.

The center of the spherical attractive force field is initially located at the PHANToM tip so the PHANToM is within the no-force region. The PHANToM has some play in this spherical no-force region, so r should be small for small position error. The force field is then shifted to the next recorded position. As this is done the PHANToM is moved out of the no-force region. The driving force (1), proportional to the distance $v–r$, acts on the PHANToM and attracts it to the zero force region of the shifted force field. The force field is then shifted to the next recorded position and this loop repeats until the end of the playback file is reached.

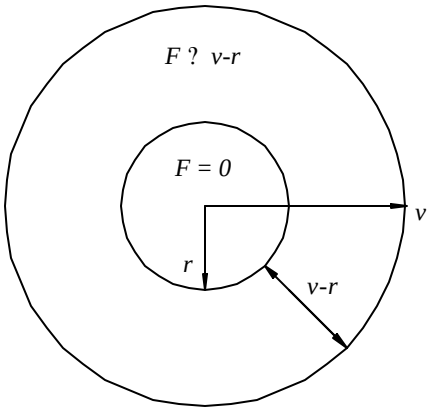

Figure 2. Spherical Attractive Force Field for Position Playback

Force Playback. During the recording mode all the PHANToM reaction forces are saved to a data file in binary mode. To achieve force feedback during playback, a force field of sufficiently large volume to cover the volume of the haptics-enabled region on the screen is created. The reaction forces are then read and sent to the PHANToM to get the playback force.

4. PLAYBACK EVALUATION RESULTS

This section presents the playback experiments and the results obtained. Four different cases were considered: with and without the human finger in the PHANToM thimble and with and without the recorded force with corresponding haptics model disabled and enabled, respectively. We will present here results for without finger case only.

For the experiment an arbitrary path of approximately one minute duration was chosen, with 41,957 path points. The path was made to interact with the virtual human skin, spine, interspinous ligaments, and the scapula. In order to compare the effectiveness of the system under different values of k and r, the same path is used for all cases.

The difference between the recorded force and position and those obtained during playback is calculated in the *X*, *Y*, *Z* directions. In this article, a mean square error (*MSE*) measure is used for both force and position:

$$MSE = \frac{\sum_{i=1}^{n}\sqrt{(X_{iR} - X_{iP})^2 + (Y_{iR} - Y_{iP})^2 + (Z_{iR} - Z_{iP})^2}}{n} \qquad (2)$$

X_{iR} is the recorded and X_{iP} the played back X component of position at the i^{th} point; the Y and Z terms are defined in a similar manner. *MSE* for force is defined analogously to (2). We also calculate the standard deviation to give a measure of the spread of position and force errors over the playback path.

Position Playback The results show that smaller r and larger k tend to yield lower position errors. But we cannot reduce r and increase k indefinitely as this introduces a buzzing effect. For the results of Figs. 3, a nominal constant value of $k = 0.38$ *N*/*mm* was used, and r was varied by steps of 0.10 *mm*. In the results of Figs. 4, a nominal constant value of $r = 0.06$ *mm* was used, and k was varied by steps of 0.02 *N*/*mm*. In both the plots, standard deviations are included for the case where the playback forces are included (shown in solid blue). Standard deviations are not included for the cases without the recorded forces played back (but with haptics model on, shown in dashed green) simply because the plots are too cluttered. The *MSEs* can be easily compared in the plots.

Using OpenGL, two curves were drawn, the red one representing the recorded positions and the green one showing the path traced by the PHANToM upon playback (see Fig. 5). It was observed that the two lines were close throughout the entire motion, so the position error is constant throughout the trajectory.

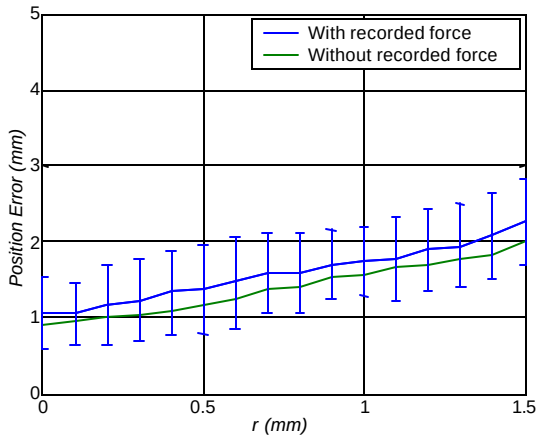

Figure 3. Position *MSE* vs. *r*, Without Finger

Force Playback. During playback the reaction forces on the PHANToM are compared with the recorded forces and the force mean square error is calculated, similar to the position *MSE* in (2); also, standard deviation is calculated and displayed as in the previous position error plots

The force error obtained is on the order of 0.3 – 0.4 *N* (See Figs. 6 and 7). For the result of Fig. 6, a nominal constant value of $k = 0.38$ *N*/*mm* was used, while in the results of Fig, 7 a nominal constant value of $r = 0.06$ mm was used.

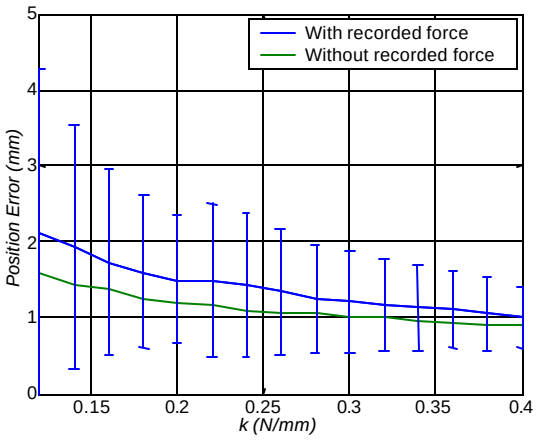

Figure 4. Position *MSE* vs. *k*, Without Finger

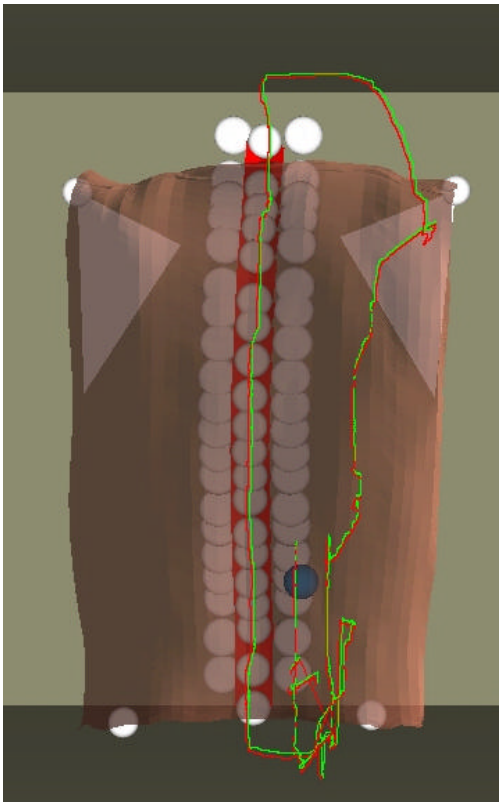

Figure 5. Playback and Recorded Paths

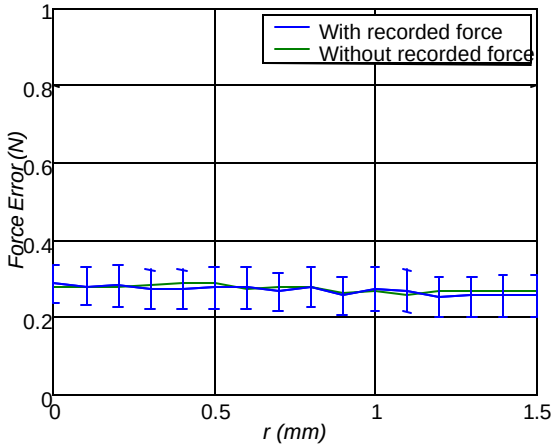

Figure 6. Force *MSE* vs. *r*, Without Finger

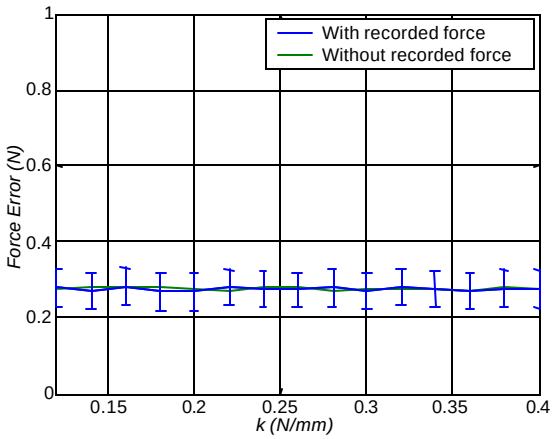

Figure 7. Force *MSE* vs. *k*, Without Finger

5. Conclusion

The position error increased with increasing *r* and decreased with increasing *k*. The position playback is faithful, observed through numerical, graphical, and subjective means. The force errors remained relatively constant, with little or no dependence on *r* and *k*.

When the system is in normal mode (not playback mode), the PHANToM motors act against the movement of human finger when an obstacle is encountered in the virtual environment. The human finger resists this force feedback to feel the modeled haptic sensations. In playback mode in our system the human finger is passive. Hence, accurate force feedback may not be obtained.

We do record the reaction forces and send them back to the PHANToM. But at the same time we send the driving force for position playback to the PHANToM, to move it through the recorded positions path. These two types of forces (position playback, haptic feedback) interact and hence the desired force feedback may not be achieved. But since the haptic forces are small, the force feedback subjectively appears to be faithful.

6. Future Work

A near-term goal in the project is to perform playback experiments with trainees to determine the potential impact of the playback feature on learning palpatory diagnosis tasks using our Virtual Haptic Back. The current article studies only the system performance regarding playback, not including any human trainees. Another future goal is to integrate two PHANToM interfaces, including playback, within the Virtual Haptic Back. Users can then use their thumb and forefinger to palpate as they do in the real world.

REFERENCES

G. Burdea, G. Patounakis, V. Popescu, and R.E. Weiss, 1999, "Virtual Reality-Based Training for the Diagnosis of Prostate Cancer", IEEE Transactions on Biomedical Engineering, 46(10): 1253-60.

M. Dinsmore, N. Langrana, G. Burdea, and J. Ladeji, 1997, "Virtual Reality Training Simulation for Palpation of Subsurface Tumors", IEEE International Symposium on Virtual Reality and Applications, Albuquerque, NM, March: 54-60.

A. Crossan, S.A. Brewster, S. Reid, and D. Mellor, 2000, "Multimodal Feedback Cues to Aid Veterinary Training Simulations", Proceedings of the First Workshop on Haptic Human-Computer Interaction: 45-49.

K.L. Holland, R.L. Williams, R.R. Conatser Jr., J.N. Howell, and Dennis L. Cade, 2002, "Implementation and Evaluation of a Virtual Haptic Back", Virtual Reality Society

T.H. Massie and K.J. Salisbury, 1994, *"PHANToM Haptic Interface: A Device for Probing Virtual Objects"*, ASME International Mech Engr Congress, Chicago, IL, DSC 55(1): 295-299.

Haptic Interaction with 3D Ultrasound Data
The e-Touch sono System

Touch Your Baby Before He or She is Born

Walter A. Aviles
Thomas G. Anderson
Novint Technologies
http://www.novint.com

Abstract

The e-Touch sono™ system allows users to interactively feel and see 3D ultrasound images. This system is currently targeted for use in pre-natal imaging [1]. Parents are able to feel as well as see three-dimensional imagery. We have found that this increases both their understanding and their enjoyment of their child's ultrasound image. This has several important benefits. First of all, it allows physicians to more clearly explain the developmental process, progress and any complications. Second, it helps ease parental stress and anxiety over the progress of their child. Finally, it helps the all-important parent-child bonding process. Over time, the e-Touch sono system will be refined in order to be used for medical diagnosis, surgical planning, and intraoperative procedures.

Background

Ultrasound is a high-quality, reliable and cost effective medical diagnostic tool. It has been used for medical imaging and diagnosis purposes since the Second World War [2][3]. The first medical ultrasound systems, mirroring their roots in Sonar technology, required that the patient be immersed in water for imaging to occur. They also used "A-mode" presentation of the ultrasound data -- blips on an oscilloscope screen. The more familiar "B-mode" presentation of two-dimensional gray scaled images followed shortly thereafter. Ultrasonic imaging came into more common use in the clinical environment in the 1960s and 1970s with the introduction of compact hand-held scanners with real-time B-mode imaging capabilities. More recently, medical ultrasound systems capable of generating three-dimensional (i.e., volumetric) data and images have become available.

Some ultrasound systems generate the three-dimensional images by analyzing and combining a series of two-dimensional B-mode images gathered using a hand-held scanner outfitted with a position/orientation tracker. Other systems utilize a more sophisticated transducer design to allow three-dimensional data to be gathered without requiring any external transducer movement or tracking. These later systems are often referred to as "4D" systems because they allow three-dimensional data and imagery to be gathered in real-time. GE Medical Systems' Voluson 730 ultrasonic imaging system, for example, currently generates 16 three-dimensional images per second [4].

One of the most common uses of ultrasound is in obstetrics and gynecology. Ultrasound has made it possible to non-invasively study pregnancy from beginning to end. Diagnosis of complications of multiple pregnancy, fetal abnormality and placenta previa became possible with the use of ultrasound in the clinical environment. Moreover, ultrasound exams are generally considered safe at the power levels used for diagnosis. In addition to providing physicians with a

useful diagnostic tool, ultrasonic imaging allows prospective parents to view their child's development.

Figure 1 -- Water Immersion Motorized B-mode Scanner Circa 1954

3D Haptic Interaction

As part of our research and development, we have explored the use of haptic interaction in the understanding, use and modification of sensor-derived three-dimensional geoscientific [5] (e.g., seismic) and medical (e.g., CAT Scan or MRI) data. 3D ultrasound is particularly interesting because it is a safe and cost effective real-time imaging modality. Adding haptic interaction to 3D ultrasound promises to have many of the same benefits and features that we have seen before. Three-dimensional haptic interaction allows for the direct and unambiguous selection of parts of the ultrasound image for further exploration/analysis, simplified and direct six degree-of-freedom control of imaging tools such as interactive clipping planes and the use of an additional sensory channel or modality (i.e., touch) for understanding the data. These features hold the promise of simplifying and improving the clinical analysis of three-dimensional ultrasonic imagery by providing both a natural and a rich additional sensory interaction mode. Clearly, however, fully exploring and understanding the clinical impact and benefits of adding touch to ultrasound images will take time.

In order to help introduce haptic technology to the three-dimensional ultrasound community, we have developed a product, known as e-Touch(TM) sono, which is aimed at providing a more informative and enjoyable experience for parents when pre-natal 3D ultrasound images are taken of their child. Parents are able to feel as well as see three-dimensional imagery. We have found that this increases both their understanding and their enjoyment of their child's ultrasound image. This has several important benefits. First of all, it allows physicians to more clearly explain the developmental process, progress and any complications. Second, it helps ease parental stress and anxiety over the progress of their child. Finally, it helps the all-important parent-child bonding process.

This initial focus on parental interaction with ultrasound allows us to introduce haptic technology into the clinical environment in a way that has clear benefits today and allows the medical/diagnostic uses of haptic interaction with 3D ultrasound to be more gradually explored.

The e-Touch sono™ System

The e-Touch sono System is a turnkey hardware and software system that allows users to interactively feel and see 3D ultrasonic images. Sono also allows the 3D images to be interactively cleaned-up and exported for the generation of 3D "hardcopy" or imagery.

Figure 2 -- The e-Touch sono System

The system consists of a computer workstation, a graphical display, a 3D touch interface device and the e-Touch sono software. Sono can also be run on a laptop computer system.

Figure 3 -- 3D Ultrasound Acquisition

The overall viewing and haptic interaction process begins when a pregnant women gets a 3D ultrasound. The data, in volumetric form, is transferred to the e-Touch sono system. The

program automatically cleans up the data and applies default visualization and haptic parameters so that the "skin" surfaces in the data are visible and touchable. The parent or physician can then see and feel the surfaces of the child's face and body. Haptic skin textures are applied to the surfaces to improve the overall haptic experience. As shown in Figure 4, modern 3D ultrasound machines allow highly detailed images to be generated. Allowing parents to interactively control this imagery as well as "feel their baby" significantly adds to the overall educational, enjoyment and bonding aspects of the experience, in addition to easing anxiety about the health of the baby.

Figure 4 -- Side by Side Images of Pre-Natal Ultrasound & Baby

The key features of the sono system are:

- Reads data directly generated by GE Voluson 730 4D Ultrasound Systems
  - Cartesian Kretz V730 volume data file format via network or CDROM
  - Other data formats to be added
- Interactive visual display of 3D ultrasound data
  - Control color and opacity in real time
  - Real time rotate, translate & zoom
- Allows user to feel the ultrasound data in three dimensions
  - Surfaces (variable feel)
  - Volume properties (variable feel)
- Allows interactive clean-up of ultrasound data for 3D export in a variety of formats
  - Volume
  - Surface
  - Point
- Generates data for 3D "hardcopy" output
  - Several processes supported
- Generates data for 3D "take home" visualization
  - Several formats supported

Technical Overview

Data Pre-Processing

Three dimensional ultrasound data, like any real-world data, is noisy. Moreover, sampling resolution is such that voxel quantization is visible in the raw data obtained directly from the 3D ultrasound machine. As a first step in our visualization and haptic interaction process, we apply a standard set of three-dimensional filters to the ultrasound data. We utilize a 3D median filter followed by a gaussian smoothing process. The parameters of these filters were empirically

tuned for use with the Voluson 730 system but need not be changed across machines or images. Figure 5 shows a typical ultrasound image before and after filtering.

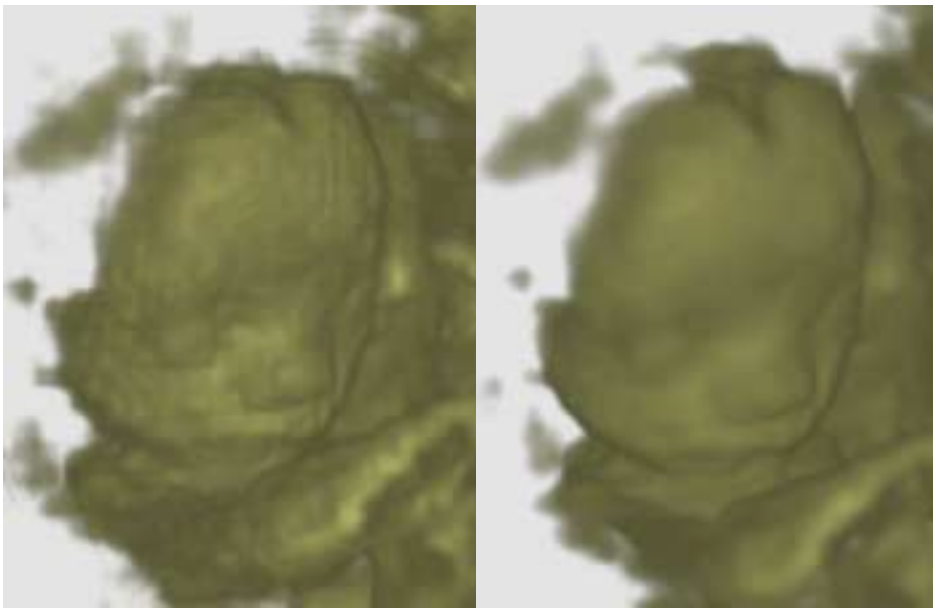

Figure 5 -- Typical 3D Ultrasound Before (Left) and After (Right) Pre-Processing

Visual and Haptic Rendering

The sono system is a volume-based visualization and haptic interaction tool. Both modalities operate on exactly the same data set in real-time. Sono can generate surface representations, however, to allow for physical ("rapid prototyping") models to be produced.

Visual Rendering. Visual volume rendering, allows internal structures to be viewed. Although this feature is not essential to the initial "parental" focus of the sono system, it is available for the exploration of the clinical diagnostic uses of the application. Volume visual rendering, being an image order technique, scales quite well with data set complexity. Modern consumer-level graphics cards provide solid interactive (i.e., greater than 10Hz) visual performance. As can be seen in Figure 5, the sono system's default visualization parameters are set so that low-density materials, such as amniotic fluid, are not visible.

Haptic Rendering. Like visual rendering, haptic rendering is also performed directly using the volume data. Two forms of haptic interaction are currently provided -- a full-volume viscosity force feedback algorithm and an isosurface rendering algorithm. The viscosity force is meant as an early tool for physicians to explore the overall volume data set. The technique generates a viscosity-like force that varies with the haptic "density" applied to a particular data set value. The isosurface rendering algorithm allows analytic isosurfaces to be felt. It is the primary haptic rendering algorithm used for parental education and bonding purposes. Both algorithms operate based only on data in the local neighborhood of the haptic interaction point. This means that overall volume size or complexity does not affect their operation.

The isosurface algorithm consists of two parts. In the first part, the isosurface is found and surface compliance forces (i.e., normal to the isosurface) are generated. This algorithm can treat even "thin shells" as non-permeable. This is important for 3D ultrasound fetal images since data

for many parts of the image (e.g., face) is often in a thin shell form. Moreover, it is desirable to be able to vary the compliance model used for the surface independent of the whether or not the surface is part of a thin shell or deeper solid structure. For fetal imaging, the surface, for example, is rendered as quite soft or compliant using a simple spring model. The second part of the isosurface algorithm, applies surface texture (i.e., forces along the isosurface). We have tuned this model to provide a "skin-like" texture to surfaces.

Surface Model Generation. The sono system can also generate surface representations of the isosurface. It currently generates stereolithography format (i.e., STL) files of the isosurface for either the entire data volume or for particular views of the data. This data can be used for manufacturing three-dimensional models of the baby for diagnostic and "keepsake" purposes.

Conclusions

The e-Touch sono system allows three-dimensional ultrasound data to be felt as well as seen in real time. The initial focus of this application is to allow visual and haptic interaction with fetal images for parental education and bonding. This particular use of the system is quite compelling and has resulted in the commercial development and distribution of the sono system. Over time, we believe that, as the potential of haptic interaction for clinical diagnostic and analysis purposes is further explored, that sono will also have an important medical role.

Acknowledgements

The authors wish to thank New Mexico Sonographics Inc. for allowing access to their ultrasound equipment in the early phases of the sono development. We also wish to thank the Novint technical team, especially Richard Aviles and Jake Jones, for their work on the sono effort and underlying technology.

References

[1] Novint Technologies Incorporated, http://www.novint.com

[2] Woo, J, *A short History of the development of Ultrasound in Obstetrics and Gynecology*, http://www.ob-ultrasound.net/history.html

[3] *History of Ultrasound*, http://www.scmobileimaging.com/history.html

[4] GE Medical Systems, http://www.gemedicalsystems.com

[5] Aviles, WA and Ranta, JF, *Haptic Interaction with Geoscientific Data*, in Salisbury, JK and Srinivasan, MA (Eds), "Proceedings of the Fourth PHANTOM Users Group Workshop," AI Lab Technical Report No. 1675 and RLE Technical Report No. 633, MIT, November 1999, pp. 78-81.

Implementing Haptic Feedback in a Projection Screen Virtual Environment

Andrew G. Fischer
Judy M. Vance[1]
Department of Mechanical Engineering
Virtual Reality Applications Center
Iowa State University
Ames, Iowa 50011
fischea1@vrac.iastate.edu
jmvance@vrac.iastate.edu

Abstract

Haptics refers to the sensing of force, weight, or other physical properties through feeling and touch. The additional information from haptic feedback makes certain engineering design tasks, such as part assembly, much easier than with a traditional computer interface. Haptic feedback, when used in a virtual reality application, is most often combined with either a stereo monitor or head mounted display. In this research, the PHANToM 1.5 haptic device is combined with a projection screen virtual environment, the C6 at Iowa State University, in order to explore the benefits haptic devices bring to this type of immersive environment. To achieve this goal several concepts are presented including: the design of a stand to support the PHANToM in the environment, a virtual volume to scale the PHANToM's physical workspace to a user defined portion of the virtual world, and an application to integrate the PHANToM's GHOST software with the vrJuggler virtual environment software. Two example uses of haptic feedback in the virtual environment are presented: a NURBS surface and a virtual assembly application. The assembly example uses Boeing's Voxmap PointShell (VPS) software to interface with the GHOST software and control the PHANToM. The benefits of haptic feedback in the virtual environment for these examples and some guidelines for using them are presented.

Introduction

A key component of virtual reality (or VR) systems is the ability to immerse a participant in a computer generated virtual environment. Immersion refers to the sense of "being there" that a user feels in the virtual world; the greater the sense of immersion, the more real the virtual world appears [1]. The level of immersion a user feels in a VR environment is related to the number of senses stimulated, such as sight and hearing [2]. However, most virtual reality systems are lacking in a key area of stimulation, namely some form of physical or haptic feedback.

One device capable of providing haptic feedback in a virtual reality simulation is SensAble Technology's PHANToM [3]. This research examines the issues surrounding the use of the PHANToM in a six-sided projection screen synthetic environment, the C6 at Iowa State University.

Implementation

Bringing the PHANToM into a projection screen virtual environment presents several challenges. Since the PHANToM is essentially a desktop device, using it in the large projection screen environment requires the PHANToM to be mobile and adjustable to accommodate a standing user. Second, the PHANToM's 19x27x37 cm physical workspace is much smaller than the projection screen environment's 10x10x10 ft size. Some method must be used to make the PHANToM useful over large portions of the virtual environment. Finally, a program must be written to integrate the operation of the PHANToM with the software controlling the virtual environment.

[1] Corresponding Author

Phantom Stand

Physically supporting the PHANToM in the virtual environment was accomplished by designing and building a stand to hold the PHANToM. This stand rolls about on four castor wheels, which may be locked to keep the stand from moving when the PHANToM is in use. Stand height is adjustable from 28 to 42 inches to accommodate different users and postures. Since knowing the orientation of the phantom stand is desirable, magnetic tracking devices, such as the Ascension Technologies MotionStar used in the C6, should be compatible with this stand [4]. Since magnetic materials adversely affect the accuracy of such trackers, the phantom stand was constructed out of bonded PVC plastic and stainless steel hardware. When the stand is in the virtual environment a magnetic tracking device is attached to one of the legs. A model of the stand appears in Figure 1.

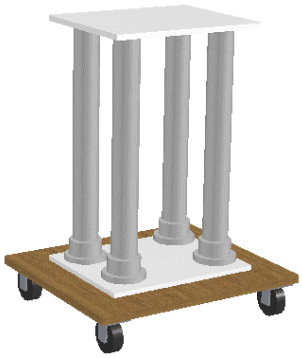

Figure 1. Phantom stand

Phantom Volume

To address the mismatch between the PHANToM's physical workspace and the size of the virtual environment, the concept of a phantom volume is presented. This consists of a user-defined quadrilateral volume of space in the virtual environment that correlates motion of the PHANToM's physical endpoint to a virtual position in the environment. This approach is similar to that taken by Preshce and Hirzinger in their work on workspace scaling for teleoperation [5].

A phantom volume is defined by selecting two opposite corners in the virtual space with a wand. The known orientation of the phantom stand, obtained from a magnetic tracker, is used to orient the volume. To aid selection, the volume is dynamically drawn between the first point and the current wand position until a second point it chosen. The completed volume may then be translated about the virtual world and/or scaled to a desired size. Figure 2 shows a phantom volume with a red sphere representing the virtual PHANToM endpoint.

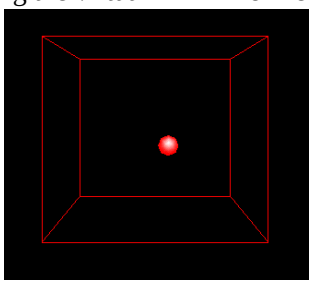

Figure 2. Phantom volume

When using the PHANToM, the virtual endpoint position is confined to the phantom volume, just as the physical endpoint is confined by the PHANToM's physical workspace. Motion of the actual PHANToM is scaled to match motion of the virtual position. This way the limited PHANToM physical working volume can be matched and used over an arbitrarily large space in the virtual environment.

Phantom Driver

The C6 is controlled with the vrJuggler software, a complete framework for virtual reality applications that may be used on a variety of virtual reality devices [6]. Integration of the PHANToM's GHOST software and vrJuggler is handled by the phantom driver program. The phantom driver is a C++ class that builds the haptic scene graph, loads the haptic geometry, positions the virtual PHANToM endpoint in the virtual environment, scales motion of the PHANToM to the phantom volume, and communicates with the rest of the simulation.

The phantom driver's first step is defining the workspace limits of the physical PHANToM device. Since the phantom volume isn't constrained to be a cube, the phantom driver creates the PHANToM's physical workspace to match the form of the phantom volume, so a single scale value suffices to match motion of the PHANToM's physical endpoint to the virtual world. This value, *world_haptic_scale*, is determined from the largest dimension of the phantom volume and

the *max_workspace_size* parameter as in equation 1.

$$world\_haptic\_scale = \frac{max\_workspace\_size}{PV[largest\_dimension]} \quad (1)$$

The *max_workspace_size* represents the largest dimension of the PHANToM's physical workspace. In some cases it is useful to confine the PHANToM endpoint to a box of modest size, ensuring that the user doesn't run out of device travel or collide the endpoint with the bulk of the physical PHANToM. At other times it may be desirable to make the physical workspace limits much larger, allowing the PHANToM free movement throughout its entire range of travel. For this work the *max_workspace_size* is set to 120.0 millimeters, which prevents the PHANToM endpoint from colliding with the physical device or the phantom stand.

The phantom driver's also must construct the haptic geometry required by the application. This geometry may consist of simple primitives and/or polygonal meshes. Once the phantom driver has the workspace set up and the geometry loaded, it uses GHOST to build the haptic scene graph. A diagram of this scene graph appears in Figure 3.

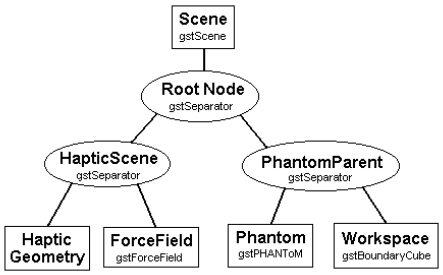

Figure 3. Haptic scene graph

A key feature of this scene graph is the placement of the Phantom and the Workspace objects into the PhantomParent separator. This allows the position of the PHANToM device and its workspace to be translated and rotated in the haptic space to match the position of the phantom volume simply by applying the proper transformations to the PhantomParent, since moving the PhantomParent separator moves both the PHANToM and it's Workspace.

Example Applications

In this research two example applications are presented that demonstrate the advantages of using the PHANToM in a projection screen virtual environment: a NURBS surface exploration example and a virtual assembly application that uses Boeing's VPS (Voxmap PointShell) software.

The NURBS surface example demonstrates using the GHOST software to build the haptic representation of an existing NURBS (Non-Uniform Rational B-Spline) surface. The geometry is then displayed in the virtual environment where the user may define a phantom volume around and interact with any portion of the surface. This application lets the user experience the additional information about an object's shape through the sense of touch while exploring the differences between the PHANToM's physical workspace size and the virtual phantom volume workspace.

The virtual assembly example has the user manipulate the PHANToM in an attempt to install a rudder pedal assembly into a simplified model for the lower front portion of a light aircraft. The haptic feedback provides a fast and intuitive way for a designer to determine if the assembly task can be completed. Boeing's VPS software is used to detect collisions between the pedal assembly and the aircraft structure. Physically-based modeling is used to calculate the haptic forces resulting from any collisions. GHOST is used to return these haptic forces to the PHANToM. The resulting program is capable of manipulating very complicated models at haptic feedback rates [7]. Figure 4 shows the virtual assembly example in use in the C6.

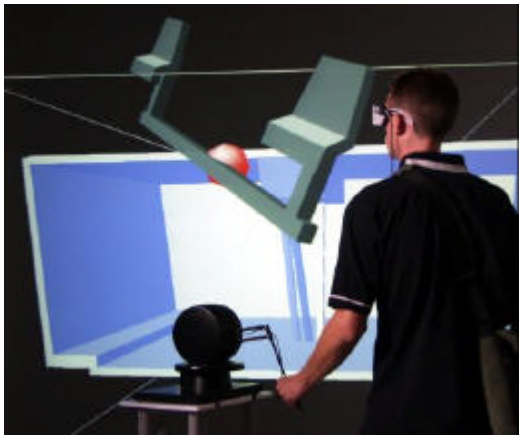

Figure 4. Virtual assembly application with PHANToM

Conclusion

After using this application and experimenting with the examples, some conclusions can be drawn about how haptic feedback and the projection screen virtual environment aid the user in different tasks.

1. Haptic feedback makes manipulating a complex part through confined spaces faster and more intuitive than using a standard keyboard and mouse approach.
2. Being able to touch objects with the PHANToM provides additional information about the geometric structure, which may not immediately be noticeable visually.
3. The projection screen virtual environment makes interference issues encountered in a particular task easy to find and remedy.
4. Since several people may observe the virtual environment and share the PHANToM in the same simulation, the projection screen environment enhances collaboration between users.
5. For the examples presented in this work, the differences in workspace size between the physical PHANToM and the virtual phantom volume appear relatively unimportant.

While the addition of haptics to the virtual environment improved the ability of users to interact with digital models for the examples presented, there are some guidelines that should be followed to ensure a high quality haptic feedback experience in a virtual environment.

1. Avoid positioning much of the phantom in a location that does not align with the user's viewpoint.
2. Avoid making the phantom volume excessively large, as this reduces the quality of the force feedback.
3. Avoid positioning the phantom stand in the direct line of the user's sight.
4. Keep the PHANToM and its stand outside of the virtual geometry.

Acknowledgements

This work was supported by NSF research grant DMII-9872604. The research work was performed using the facilities of the Virtual Reality Applications Center, Iowa State University.

References

1. Pausch, R., D. Proffitt, and G. William. *Quantifying Immersion in Virtual Reality*. in *24th annual conference on Computer graphics and interactive techniques*. 1997: ACM Press.
2. Burdea, G.C., *Force and Touch Feedback for Virtual Reality*. 1996, New York: John Wiley & Sons, INC.
3. Massie, T. and J.K. Salisbury. *The PHANTOM haptic Interface: A Device for Probing Virtual Objects*. in *ASME Symposium on Haptic Interfaces for Virtual Environments*. 1994. Chicago, IL: ASME.
4. VRAC, *About the C6*. 2001, Iowa State University.
5. Preusche, C. and G. Hirzinger. *Scaling Issues for Teleoperation*. in *Fifth PHANToM Users Group Workshop*. 2000. Aspen, Colorado.
6. Bierbaum, A., et al. *VR Juggler: A Virtual Platform for Virtual Reality Application Development*. in *Virtual Reality, 2001*. 2001. Yokohama, Japan: IEEE.
7. McNeely, W.A., K.D. Puterbaugh, and J.J. Troy. *Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling*. in *26th Annual Conference on Computer Graphics and Interactive Techniques*. 1999.

The Interchange of Haptic Information

White Paper Submitted by Novint Technologies, Inc.

Overview

Haptic interaction is applied to and important for a wide variety of fields. There is active research and development, for example, utilizing haptics in the CAD/CAM, industrial design, CGA, geosciences and medical fields. There is, however, no widely used interchange format for haptics-relevant information. More widespread research and development efforts utilizing haptics are, of course, relatively recent so the lack of widely used interchange formats is to be expected. Yet this lack of haptics information interchange formats is an impediment to the development of the field. It is difficult to communicate both "finished content" and methods/approaches. Starting to understand the issues associated with the interchange of haptic information and defining an approach to achieving this interchange could be very useful to the field. Although it is quite likely that any early attempts will be shown to be incomplete or ill directed over time, it is only by attempting to address these issues and learning from the process that commonly used haptic interchange formats will emerge.

This "white paper" discusses some of the issues related to the interchange of haptic information, outlines one possible framework for an interchange format and discusses a particular instantiation of this framework. The authors wish to emphasize that this is “work in progress”. The main goal of this paper is help to start to understand the scope of the issue and to generate meaningful discussion concerning the interchange of haptic information.

Interchange Format Standardization Background

An important factor in the success of any technological or scientific field is the evolution and development of methods to help specify and communicate core ideas and concepts within that field. In other words, some form of common “language” or notation to capture and convey key observations, notions, models, theories and other relevant information is advantageous. In reality, there is usually a collection of languages/notations/grammars that are utilized to help support research, development and utilization within a given field.

The interchange of “scene” information (i.e., objects and their characteristics) in computer graphics/animation (CGA), CAD/CAM and virtual environments is one important example of the use of notation and grammar to specify and communicate technological information. Geometric and visual rendering information is communicated via commonly accepted file formats. This allows compact interchange of scene information between users and, oftentimes, the utilization of this data across applications. Benefits include: a) support for scene persistence; b) tools created by different organizations can be used to refine and render the scene; and c) representational models and characteristics are made explicit.

Graphical interchange formats for three-dimensional data include proprietary and vendor-derived formats. Many of these have such widespread use that they have become "de-facto" standards.

These include widely used vendor-specified formats such as Alias Wavefront (i.e., *.obj), 3D Studio (i.e., *.3ds) and AutoCAD (i.e., *.dxf). Other graphical interchange formats are the result of actions by standards committees in defining new industry-wide formats or refining vendor-specified formats into more "vendor neutral" forms. Examples of such standards in three-dimensional computer graphics, animation and CAD/CAM include IGES, VRML 1.0/97/2.0 and, more recently, X3D. All of these file formats allow the specification of three-dimensional geometry. Most of these file formats allow specification of visual rendering relevant parameters such as lighting, color and texture. VRML and X3D also allow hyperlink and behavioral information to be encoded.

As can be seen, even a relatively mature field such as computer graphics has a plethora of vendor-specified and "standards based" information interchange formats. The most successful of these tend to be "de-facto" standards or are based on de-facto standards. This is a natural consequence of the maturation of a field. Initially, since no other alternatives exist, specific vendors or research groups develop interchange methods for their own use. The utility of these interchange formats is assured as the formats are refined and extended within the context of real applications of relevance to the field. As the programs that utilize these methods become more widely used, the particular interchange format becomes of interest to other vendors and research organizations. Typically these other groups will directly utilize or provide conversion utilities to the interchange format in order to reach the community that is using the interchange format. In time, the interchange format may be the basis for (or at least an important component in) industry/field "standards" efforts. Typically, the more "open" that the original interchange formats are, the more quickly that they evolve, the more quickly that they are adopted and the more widely they are used by the community.

Competing interchange formats and standards, of course, emerge. This tends to be healthy for the field in that different interchange formats oftentimes tend to address different/more specific needs in the field. They also tend to be informed by or build on previous efforts so that refinement and evolution can occur relatively rapidly. It is sub optimal, however, if these competing interchange formats tend to address the same needs and requirements. Sometimes standards efforts are undertaken when there is too much overlap and fragmentation in interchange formats.

Haptics Information Interchange Requirements

What kind of information is required to support haptic interaction? The answer to this question depends upon the overall application domain, the characteristics and limitations of haptic devices, the methods used to control these devices and human perceptual “haptic percepts”.

Field of Use Issues

Different fields and applications emphasize different aspects of haptic rendering and interaction technology. Many manufacturing CAD/CAM applications, for example, would be well served with the ability to haptically render rigid solids and to spatially manipulate them. Industrial design applications tend to put more of a premium on rendering of surface properties such as compliance and texture. The rendering of deformable bodies is important to many applications in the medical domain. Even among these few example domains, a variety of underlying geometric models are encountered -- both surface and volumetric representations. Applications and domains with more abstract data, such as computational fluid dynamics and geoscientific data

processing, do not even utilize classical geometric models. These domains tend to emphasize haptic rendering and interaction with field data and abstract/derived/computed spatial properties. Finally, other domains, such as CGA, utilize and interchange information related to the dynamic behavior of objects within scenes.

Device and Control Issues

Data interchange formats and approaches are also informed and constrained by the rendering models/approaches and technologies that are used. In computer graphics, for example, scene geometry is often supplemented by information to support widely used visual rendering approaches. Color information (often using a multi-component model) is provided along with lighting information. It should be noted, that like geometric models, even "simple" information such as color can be specified in a variety of formats (e.g., RGBA, CMY, etc.). These options exist even in a field where the rendering technologies and devices are relatively mature and "standardized" (e.g., CRT displays and LCD displays).

In the haptic rendering and interaction domain, the devices and the underlying control methodologies/approaches are anything but standardized. For example, the number of degrees-of-freedom supported by haptic interaction devices for "kinesthetic feedback" can range from one or two degrees of freedom for more "gaming oriented" devices (e.g., Logitech force feedback mouse) to dozens of degrees of freedom for some special purpose and research devices (e.g., SARCOS dexterous arm master). Control methodologies range from simple "open loop" position/force control approaches (typically utilized with devices such as the PHANTOM haptic interface device) to "closed loop" bi-lateral servo control systems where forces are directly sensed and controlled (e.g., SARCOS dexterous arm master).

Tactile feedback devices or components come in at least as many configurations as their kinesthetic feedback cousins. These devices may be able to be roughly categorized by such characteristics as area of influence and density of "stimulator" elements but there is little standardization beyond that. For example, some devices utilize electro-cutaneous stimulation techniques and others utilize mechanical vibration or deformation techniques. So even the underlying manipulated state variables are different.

Perceptual Issues

Perceptual models also enter into what kind of interchange information is required for haptics. The sensory capabilities of our visual systems, for example, are reflected in some color representation techniques used in CGA. Although much is known concerning human haptic capabilities, a commonly accepted haptic "perceptual language" has yet to emerge. Very broad haptic discrimination categories such as shape, compliance and texture, however, can help inform the development of a haptic information interchange approach.

Other Efforts

A significant percentage of software research and development efforts utilizing haptics define or otherwise use some representation of the visual and haptic attributes of the objects that they employ. Whether the representation is implicit or explicit, some representation of the haptic properties of objects is used in order to support scene and program persistence. To date,

however, little of this work has entered the public domain. For the most part, these efforts are project/institution specific or proprietary. An example of a proprietary approach may be found in the ReachIn Technologies Magma API. Specific extensions to the Virtual Reality Modeling Language are used in Magma to capture haptic information in a form that is consistent with the haptic rendering approach used by that company. The details of the haptic rendering algorithms themselves are not in the public domain. There is some initial activity, however, in the public standards sector. CSIRO has proposed extensions to the MPEG-4 standard. These extensions, however, are based on Magma work and suffer some of the same limitations – they assume and "prescribe" a specific set of rendering capabilities but do not specify the details of the algorithms themselves.

Haptic Information Interchange Framework

Given the state of haptics research and development, is it even meaningful to discuss haptic information exchange methods and formats that could be widely applicable? Can we do more than simply specifying methods for capturing haptic-relevant control variables and information for use by proprietary or highly simplified algorithms?

We believe that the answer to these questions is yes. Our approach is to go beyond "prescriptive" approaches and develop an extensible descriptive framework that maps well to existing haptic rendering approaches while maintaining structural flexibility to grow over time.

The key aspects of the haptic rendering and interaction domain that must be taken into account in the haptic information interchange framework are:

- ?? Model of the Haptic Rendering Process
- ?? Spatial and Behavior Data to be Rendered
- ?? Model of the Haptic Device and Associated Control Mechanisms

The remainder of this paper provides an overview of a particular haptic information interchange framework and discusses a specific implementation of that framework utilizing X3D.

Modular Haptic Rendering Chain

A fundamental observation in developing haptic information interchange framework is that haptic rendering of a wide variety of data types and behaviors can be modeled as a connected series of processing steps or modules. Previous processing modules in the haptic rendering chain potentially inform each module in the chain. Any given module may also affect the data.

For example, in allowing haptic rendering and interaction with a dynamic rigid body, one typical approach is to compute the location of the surface relative to the haptic interaction point and use this information to compute surface compliance (i.e., a force in the direction of the surface normal). When surface texture is computed, the direction of the compliance force can be used to filter texture information so that it is to be applied tangent to the point of intersection. To compute dynamic behavior for the object, the results of the previous computation steps are used as the basis for computations that alter the position and orientation of the object. Figure 1 shows this particular rendering chain.

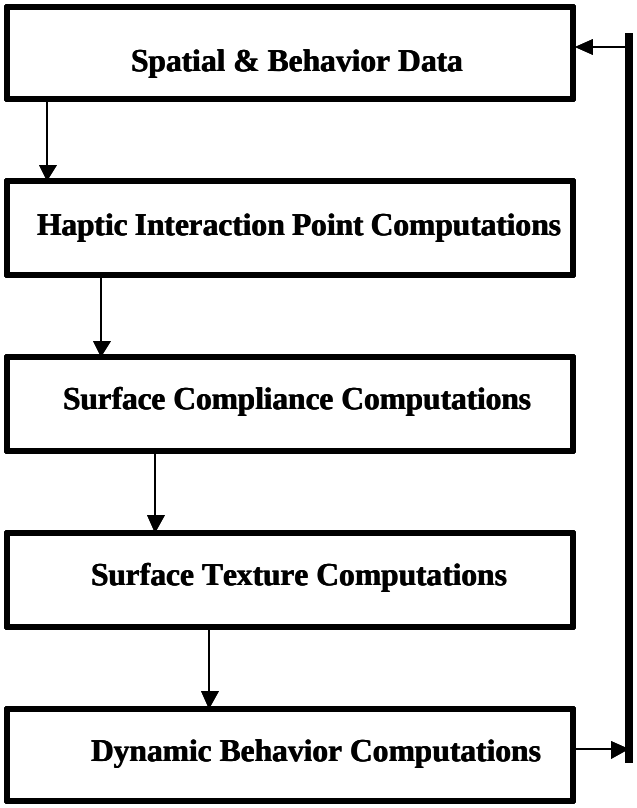

Figure 1 – Haptic Rendering for Dynamic Rigid Body

A more abstract representation of this "modular haptic rendering chain" (MHRC) is given in Figure 2. Since the actual algorithms utilized in any given step or module can vary within and across domains, the framework is designed to allow module specific rendering code to be specified in the interchange format itself. In other words, allowing the interchange format to provide the actual rendering code for modules supports portability and extensibility.

In addition, the MHRC framework allows for specification/designation of module algorithms by named reference. This allows module actions to be specified by referencing a commonly accepted/available rendering algorithm (as these become more common). This same mechanism can be used to reference more proprietary approaches. As the field matures, it is expected that the key rendering steps for given domains will become more "standardized" and that the framework could specify a default "common" algorithm to use as well as a "proprietary" algorithm to use when it is available.

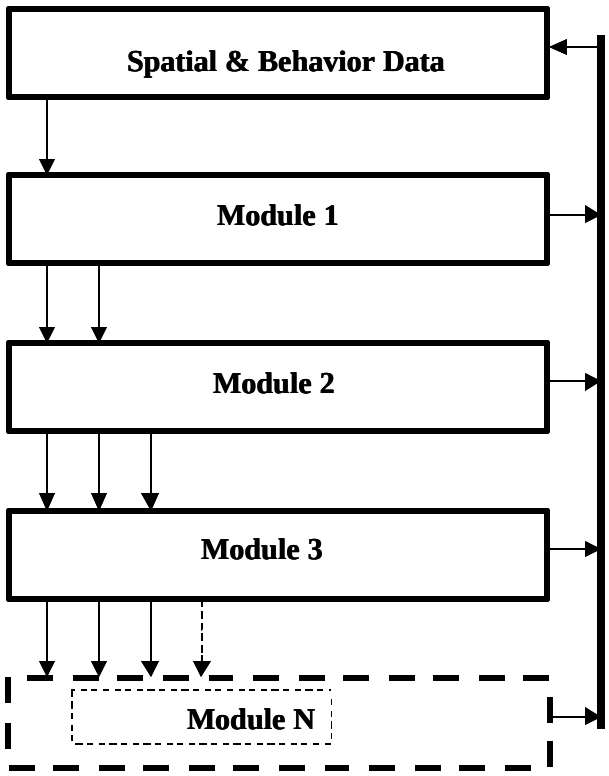

Figure 2 – General Modular Haptic Rendering Chain

The MHRC approach not only allows for abstraction of the haptic rendering chain but it also provides a way to deal with issues associated with different haptic device capabilities and control mechanisms. If information up and down the module chain is viewed as state vectors then varying degrees of freedom can be efficiently encoded. With proper abstraction, this mechanism might allow devices with differing degrees of freedom to use the same module chain. Devices with fewer degrees of freedom could just ignore non-relevant parts of the state vector used in carrying "computed forces". For example, a device that has only three positional degrees of freedom could ignore force information relating to rotational degrees of freedom. At this time, different control approaches are envisioned to be handled within the modules themselves.

The spatial and behavior data that is encapsulated in the MHRC framework, of course, domain and applications specific. Some applications, for example, will utilize polygonal data, others rational B-splines and yet others will utilize volumetric data. The MHRC framework allows this data to be encoded in the interchange specification. Initially, different rendering chains will be used for different data types. It is felt that in time, however, more general haptic rendering techniques can be developed and all that may be required to utilize them is a "data pre-processing" step.

A Test Implementation

Figure 3 – The Layout Application

Clearly there is significant clarification and refinement that must occur for the MHRC framework to prove its utility. As a first step and in order to provide a concrete test of the overall concepts, a dynamic rigid body application utilizing the MHRC approach is being developed. This application, known as Layout (Figure 3), allows rigid bodies, modeled as polygonal data to be instantiated. These objects exhibit surface compliance and texture. These objects can be haptically manipulated and exhibit dynamic behavior in the form of physically based simulation of rigid body motion. Constraints can also be placed on object motion. In order to test the feasibility of utilizing and building on existing graphical interchange standards, the MHRC interchange format instantiation is based on the X3D standard. The X3D standard essentially captures the VRML specification within an XML syntactic framework. XML is extensible so we were able to "piggy back" the MHRC components using X3D itself. This was done in such a way that a haptically enabled scene can still be viewed using a standard X3D "graphics only" viewer. In addition, XML and X3D allow us to specify MHRC module algorithms directly. Appendix 1 provides additional specific information on the Layout test implementation of the MHRC framework

Summary

The specification of a method and approach to the interchange of haptic information is a challenging endeavor. The relatively early state of the haptic rendering and interaction field, coupled with a wide variation in the types of data that must be rendered and the devices available to render them, makes the specification of a "prescriptive" haptic information interchange format sub optimal. We have proposed a way to start to organize and specify haptic rendering and interaction known as the Modular Haptic Rendering Chain (MHRC). This approach is "descriptive" in nature – it provides a framework for embedding a variety of haptic rendering approaches in a structurally coherent manner. In order to better understand and test the MHRC concept, a haptically enabled dynamic rigid body application, known as Layout, has been developed and is being refined. Although the MHRC approach is clearly imperfect and requires significant refinement, we believe that it is a solid "stake in the ground" and bears further exploration.

Appendix 1 – Test Implementation

MHRC in a Dynamic Rigid Body Application

The Layout application provides an initial test implementation of the Modular Haptic Rendering Chain (MHRC) framework. As previously discussed, this application allows rigid bodies, modeled as polygonal data to be instantiated. Haptic rendering of texture and surface compliance are supported. Objects can be haptically manipulated and exhibit dynamic behavior in the form of physically based simulation of rigid body motion. Constraints can also be placed on object motion. For this application, the MHRC framework is instantiated using X3D formalisms. Parsing of algorithms for MHRC framework modules has not yet been added. This particular instantiation of the MHRC framework is referred to as the Multimodal File Format (MMFF) in this document.

X3D Encoding

X3D Features Used

The minimum set of required X3D nodes is:

- ?? Transform
- ?? Group
- ?? Geometry
- ?? Shape
- ?? IndexedFaceSet
- ?? Appearance
- ?? Material
- ?? Color
- ?? Coordinate

In addition, support for prototyping is needed. Other node types and X3D functionality can be included in the file, and should be handled but do not need to be implemented. The interactive aspects of X3D are not (initially) needed. These should be parsed by an application, but may be ignored.

X3D content can be authored in a variety of data encoding formats, including XML encoding format and VRML encoding format. The examples throughout this document use the VRML encoding format, since it's easier to read. In practice, however, the Multimodal File Format uses the XML encoding format. The XML encoding of the examples used in this document are provided in digital format along with this document. These files were generated using the X3D-Edit program provided by the X3D Consortium. The current (16 February 2002) version of X3D Document Type Definition (DTD*) x3d-compact.dtd* was used.

Object

An Object node will describe each object within the scene. The Object node will specify:

- ?? Position
- ?? Orientation
- ?? Geometry
- ?? Graphic appearance
- ?? Haptic Properties
- ?? Behaviors
- ?? Bounds

Appearance, Transform and Bounds

Nodes to describe appearance, transform information, and a bounding box are already in X3D, and we continue to use them within the Object description. The Object will extend the X3D Shape node:

```
Object {
        position
        orientation
        geometry
        appearance
        behaviors[]
        bounds
}
```

The HapticAppearance node will extend the X3D Appearance node by adding a surface field, which is filled by a Surface node:

```
HapticAppearance
{
        material
        texture
        texture_transform
        surface
}
```

The Surface node is described in the next section.

Haptic Properties

A Surface node will specify the forces generated while touching the object. It is composed of a Compliance node and a list of Force Modifiers.

The Compliance node will specify the algorithm used to calculate the initial surface contact force. A default SpringForce node will be provided, which will generate the force using a spring model and the tools distance from the object surface. A different force algorithm can be applied by setting a different Compliance node.

The Force Modifiers filter the initial force, to specify additional haptic properties of the surface (e.g., friction, texture maps, etc.). Modifiers will take as inputs the Object's state, the initial force, and the force after all previous modifiers have been applied. Each modifier will provide fields for all of the additional information needed to describe it. If two modifiers added to a surface have the same fields, the information will still be specified in each Modifier. For example, many Modifiers may have a spring constant field but it will still be specified in each Modifier.

Ordering

The order in which Modifiers are applied may affect the output. Two levels of ordering are provided. Each Modifier will have a priority field indicating the order in which it is applied. Modifiers with the same priority are applied in the order they appear in the file. Each Modifier type will have a default priority, to specify the preferred ordering; the priority can be changed by filling in the order field. It is expected that for most cases the default order will suffice.

Priorities of 1-10 are allowed. Any Modifier that doesn't require an input force will have a priority of 1. Modifiers that should always be applied last will have a priority of 10.

An example Surface:

```
Surface
{
        compliance: Spring {     stiffness 500 }

        modifers [
                BumpMap{
                        texture: {....}
                        height: 0.5
                        priority: 2
                }

                Damping {
                        ratio: 0.2
                        priority: 10
                }
        ]

}
```

Behaviors

Behaviors modify the Object's state, but do not calculate any forces. They may take the final force calculated from Surface as an input. Dynamics, and Constraints will be specified as behaviors. For example:

```
Object{
        behaviors[
                PlaneConstraint{
                        normal 0 1 0
                        distance 0.5
                }
        ]
}
```

X3D Integration

Several additional node types have been specified. X3D allows new node types to be specified using the PROTO statement. Each prototype is derived from a parent type, and the prototype can be used in place of a node of the parent type.

PROTO declarations for the extra node types specified are shown in figure X. (haptic.wrl). The Object node is constructed from a Shape node, with an integrated transform to provide position, orientation and bounds. A HapticAppearance node extends the Appearance node by adding a field for a Surface node. The Surface node is a child node, and contains fields as described earlier. Behavior nodes subclass Child Node, so they can be added to a Group.

The relationship between Object, Surface, Modifiers and Behaviors will not be described in the file with X3D routes, it is implied by the node types.

Reading From a Standard X3D browser

Each of the new nodes, including each type of Modifier and Behavior, will be defined with a PROTO statement. All of these prototypes can be collected in a single file, each and can then be referenced from data files using the EXTERNPROTO statement. This allows the data file to be parsed by a standard X3D browser. The new node types will be "inert" when read into a standard browser.

A short example.

haptic.wrl

```
PROTO Object{…}
PROTO HapticAppearance{…}
```

data.wrl

```
EXTERNPROTO Object "Haptic.wrl#Object"
EXTERNPROTO HapticAppearance "Haptic.wrl#HapticAppearance"

Object{
        appearanceHapticAppearance {}
}
```

Extensions

The X3D spec allows for PROTO nodes to be implemented in an extension (i.e., programming) language and included in the scene.

The haptic renderer could be extended to support this capacity for Modifiers. New modifiers could be implemented as a Java™ class, and added to the scene through X3D's externproto syntax. For example:

```
EXTERNPROTO MyModifier "MyModifier.class"
```

In this way new surface Modifiers can be implemented and used with no need to update the haptic renderer. New behaviors could be implemented in the same way.

The Java™ classes for the new modifiers would be distributed with any scene file that used them.

Example

As an illustrative example, two files, in VRML format, are provided. The *haptic.wrl* file contains all the Proto definitions that are needed to allow a standard X3D browser to parse the files.

The *sample.wrl* file contains a scene with a single Object. The data file (i.e., *sample.wrl*) uses ExternProto nodes to get the definitions of each new node type from the *haptic.wrl* file. A sphere Object is specified. Two "force filters", a BumpMap and a Force Anchor, are added to the surface forces. A constraint and gravity are added to the behaviors

Haptic.wrl

```
PROTO PlaneConstraint{
        field SFFloat distance 0
        field SFVec3f normal 0 1 0
        field SFBool magnet false
}

PROTO Surface {
        field SFNODE compliance
        field MFNODE modifers
}

PROTO HapticAppearance{
{
        DEF APP Appearance {}
        EXTERN APP     # Export causes HapticAppearance to be derived
                                from Appearance

        field SFNODE surface
}

PROTO Object
{
        field SFVec3F position
        field SFRotation orientation
        field MFNODE behaviours
        field SFFloat scale

        DEF TRAN Transform
        {
                children DEF SHP Shape { }
        }

        position TO TRAN.position          # Route fields
        orientation TO TRAN.rotation

        EXPORT SHP   #Export causes Object to be derived from Shape
}
```

Other PROTO's for Modifiers would also be specified in this file.

Sample.wrl

```
#include prototypes from haptic.wrl file.
EXTERNPROTO Object [] "http://.../haptic.wrl#Object"
EXTERNPROTO HapticAppearance[] "http://.../haptic.wrl# HapticAppearance "
EXTERNPROTO PlaneConstraint[] "http://.../haptic.wrl# PlaneConstraint "
EXTERNPROTO ExternalForce [] "http://.../haptic.wrl# ExternalForce "
EXTERNPROTO BumpMap [] "http://.../haptic.wrl# BumpMap "
EXTERNPROTO ExternalForce [] "http://.../haptic.wrl# ExternalForce "

# Add an object

DEF MY_SPHERE Object {
        position 0 1 0

        apearance HapticAppearance {
                geometry Sphere {}

                material Material{}

                surface Surface {
                        compliance SpringForce {stiffness 500 }

                        modifiers [
                                BumpMap {
                                        texture ImageTexture {}
                                        height 0.5
                                        initial_force TRUE
                                }

                                AnchorSet {
                                        anchors AnchorPoint3D {
                                                forceToMove 5
                                                }
                                }
                        ]
                ]
        }  # end appearance

        behaviours[
                PlaneConstraint {
                        normal 0 1 0
                        distance 0.5
                }

                DEF GRAVITY ExternalForce {
                        force 0 -10 0
                }

        ]
}
```

Response time consistency of the GHOST force loop

A. E. Kirkpatrick and Jason Sze
School of Computing Science
Simon Fraser University
Burnaby, BC, V5A 1S6 Canada
{ted,jszea}@cs.sfu.ca

Abstract

The consistency with which Windows 2000 invokes the GHOST force loop was measured on a 900 MHz machine. The median loop time was accurate at 1 ms. However, as the computation in the force loop increased to 500 ms, the consistency of the loop decreased up to 25%. Inaccuracies in loop times were also introduced by higher network load.

Introduction

The quality of the force display is an important factor in haptic applications[1]. The device, control algorithms, and application program all contribute to the quality of the display. All these components run in cooperation with a silent partner, the operating system kernel. The kernel provides the environment in which the device driver runs and the low-level facilities by which the driver communicates with the hardware. It provides the file system, network, interprocess communication, and virtual memory facilities upon which the application is based. Most importantly, the kernel scheduler determines when the application will refresh the force and graphic displays and the world model.

While much work has been done to locate and eliminate inadequacies in force display, control algorithm, and application design, we are not aware of any research on the effect of the operating system algorithms on the performance of haptic systems. To date, researchers and application programmers alike have presumed that the operating system is "good enough". In this paper, we consider the effects the operating system scheduler might have on this performance. We begin by considering the possible mechanisms by which the scheduler might impact system performance. We point out that an important measure of system performance should be the distribution of response times for the force refresh loop. We then present some initial measurements of the response time distribution for a particular PHANTOM configuration, a single-processor system running Windows 2000 and GHOST 3.1. We conclude with a discussion of the implications of these results for haptic applications.

---

[1] In a talk given at PUG '01, the first author emphasized the importance of separating our terms for display technologies—forces and graphics—from the haptic and visual systems, the human perceptual systems interpreting those displays. In keeping with that principle, we shall refer to "force displays", the "force rendering loop", and so forth. However, given the established usage of such broader terms as "haptic application" and "haptic programming", we retain them here.

The operating system scheduler and the structure of a haptic application

Haptic programming is inherently multi-threaded. The classic structure for haptic applications consists of two loops, one computing the force display based upon the current cursor position and the location of objects in the world model, the second computing the graphic display based upon the same information. Because the human visual and tactual receptors have differing response rates, these loops run at different rates. Currently, SensAble's GHOST software architecture sets the force loop at 1000 Hz. Graphics loops typically run between 10 and 40 Hz. More importantly, because the consistency requirements of the tactual mechanoreceptors are more stringent than for the visual receptors, the force refresh loop is typically run at a higher priority than the graphics loop.

An important but little discussed consequence of this multithreaded architecture is that it makes the operating system an inherent component of the application, with operating system scheduling algorithms limiting the application's quality of service. The application (or, in the case of GHOST users, the application framework) may request a theoretical rate of force display but it is the scheduler that determines the actual rate. This scheduler is itself a complex algorithm, particularly when considered in terms of its interactions with the other services provided by the operating system. Consequently, it is imperative for haptic programmers to have clear descriptions of the limits of the scheduler they are using. In this paper, we begin developing such a description.

The scheduler as a source of noise in the force signal

The fundamental force computation is a read-compute-write loop. For each tic of the clock, the application reads the current location of the cursor, computes forces at the tip based upon its interactions with nearby objects, then writes the forces back to the motors of the display. The scheduler determines when each iteration of the loop occurs.

There are two possible kinds of noise the scheduler can introduce into the force signal. First, it can invoke the force calculation consistently slower or faster than the stated rate. We refer to this effect as *drift*. Second, the time between successive invocations may vary. We call this effect *spread*. We note that some degree of drift and spread is inevitable. The important question is their magnitude.

Spread and drift have several consequences. Irregular invocations of the force loop can create irregularities in changes to the displayed force. If the application assumes time between loop iterations is constant, timing spread will produce inaccurate force computations. This is potentially most severe for algorithms that use higher-order terms such as velocity and acceleration. Longer delays between force computations can produce large force changes when the cursor is penetrating a rigid surface. Extreme force changes can cause the drivers to shut down the computation. Finally, irregular force computations can produce inaccurate high-frequency forces. In the worst case, small surface features may be omitted altogether.

The ultimate metric of the above effects is psychophysical: Does the human user perceive the irregularities in a given application? However, such a metric incorporates far more than the operating system effects. For the purpose of measuring the specific contribution of operating system delay, we instead directly timed the invocations of the force loop.

Method

Timing loop consistency was measured using a skeletal GHOST application. The virtual world consisted of a single object located at the origin of the coordinate system. The object had a bounding volume far larger than the PHANTOM working area, causing its collisionDetect () method to be called for every invocation of the force loop. The collisionDetect () routine stored the time of its invocation in an array. Time was recorded using the Windows HighPerformanceTimer facility. This timer operates under ten μs accuracy, more than good enough for accurate measurement of the one ms force loop. The collisionDetect () routine optionally executed a busy delay loop. The delay simulated varying degrees of force computation. The collisionDetect () routine completed by returning zero, indicating no collision with the object.

The body of the application consisted of a graphics loop executed by a Windows timer interrupt every ten ms. This loop also recorded its execution time but this data is not reported here. The graphics loop displayed the PHANTOM cursor as an OpenGL sphere but did no other rendering.

This sample application was run for one minute. The PHANTOM was not touched during this time. As there were no objects for the tip to contact in this virtual world, moving the PHANTOM would have had no effect on the results. After the minute elapsed, the differences between the 60,000 successive force loop invocations were computed and written to a file.

Timings were collected under various conditions. In the unloaded condition, no other applications were active. In successive runs, the force loop delay was varied from 0 to 700 μs in increments of 100 μs, with a final run of 750 μs. In the loaded condition, a network-intensive application was run simultaneously. The network application opened up ten TCP connections and received an 888 byte message from each connection every ms, for a total throughput of 8.9 Mb/s. This application created conditions of high network load. The GHOST timing application was run concurrently, again with force loop delays ranging from 0 to 750 μs.

All timing runs were performed on a Windows 2000 (release 5.00.2195, SP2) system with GHOST 3.1. The hardware was a single-processor AMD 900 MHz Thunderbird with 500 Mb of PC-133 SDRAM and an nVidia GeForce2 card.

Results

To measure the extent of drift and spread, the .01, .05, .50, .95, and .99 quantiles were computed for the 59,999 elapsed loop times for each run. For all runs, the median (.50 quantile) was 1.00 ms. This configuration of Windows has effectively no drift under the tested conditions.

To .01, .05, .95, and .99 quantiles are a good measure of the spread of the loop times. The inner .05 to .95 band indicates the range of 90% of the loop times. A further 8% of the loop times lie in the outer band, outside the .05 -.95 range but within the .01 and .99 quantiles. Given a uniform distribution of these times, on average one loop every 80 ms (12.5 Hz) would fall in the outer 8% band. This frequency is within the detection range of human mechanoreceptors.

Figure 1 shows a plot of the .01, .05, .95, and .99 quantiles for the unloaded condition (solid lines) and the loaded condition (dashed lines). The 90% band is tight, ±.03 ms. While the 90% band is constant irrespective of the loop delay, the 95% band widens as the computation load of the haptic loop increases, from ±.08 ms at the 0 ms delay to a negatively skewed interval (0.66 to

1.18 ms) at the 500 ms delay. Beyond 500 ms, the lower limit of the loop time becomes bounded from below by the loop delay itself and the range shrinks, although the upper limit remains high.

The loaded condition has a wider spread than the unloaded condition but is much less affected by increasing loop delay. The 90% band is wider (±.08 ms) than for the unloaded condition. The 95% band is relatively constant with a more symmetric interval (.72 to 1.21 ms) of about the same width as the unloaded condition. Unlike the 95% band for the unloaded condition, however, the loaded band is basically constant across the range of delays, with only a slight decrease in the lower limit.

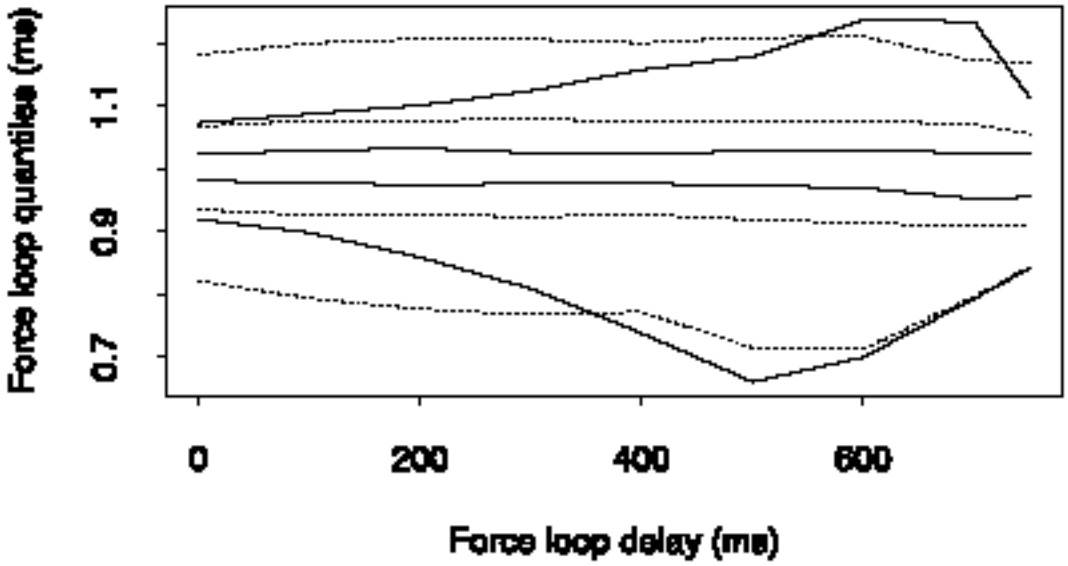

Figure 1: .01 (lowest line), .05, .95, and .99 (highest line) quantiles for unloaded (solid lines) and loaded (dashed lines) conditions at various levels of delay.

Conclusions

This system configuration exhibited generally stable performance in the consistency of its force loop. However, even for conditions of minimal load, there was sufficient spread to the response times to have humanly perceptible effects. Developers of haptic applications with exacting performance standards may wish to account for instabilities introduced by the operating system scheduling algorithms. Of course, the range of haptic configurations is vast. The operating system and GHOST release used in this initial study have both been superceded by more recent versions. Multiprocessor configurations will show less direct interference but synchronization of memory accesses between the processors may introduce new problems. In general, we suggest that the operating system scheduler will have potential impact on the performance of haptic applications for typical configurations of the foreseeable future. We plan to extend this work to measure the performance of more elaborate configurations. We also plan to examine the psychophysical consequences of the variability introduced by the operating system.

Acknowledgements

Funding for this work was generously provided by NSERC and a Simon Fraser University President's Research Grant.

Touch&Tell: A game-based tool for learning to use the PHANToM

Eric Acosta, Bharti Temkin PhD
Department. of Computer Science, Texas Tech University
Lubbock, TX 79409
Bharti.Temkin@coe.ttu.edu

Abstract

We have developed a game, Touch&Tell, with the objective of teaching how to use the PHANToM. The game starts with simple 3D shapes and leads to progressively more complex 3D objects, including anatomical structures. The purpose is to overcome some of the problems encountered by inexperienced users, such as excessively large hand movements and loss of the object contact, and to increase the user's sensitivity to computer generated sense of touch. Visual and audio cues are initially included in order to help locate and stay in contact with the object; the cues can be removed as the skill improves.

The Touch&Tell game can be also used to collect data that elucidates decomposition of the sense of touch into sensory components. To this purpose the 3D object can be made invisible in order to help identify the effects of the visual component of haptics. Haptic skills can be monitored and recorded and progress evaluated for different levels of complexity. This makes it also possible to quantify the effectiveness of point-based haptic devices.

1. Introduction

Haptic sense, based on distributed and kinesthetic receptors, provides information about objects and surfaces being touched [1,2]. The sense of touch is a complex phenomenon involving many senses and the perceived sense of touch is subjective and dependent on physiological and psychological factors. For example, human psychophysical experiments have shown that both haptic and visual systems can create biases in virtual environments when used alone, but compensate for each other when used jointly [3]. Visual information can alter the haptic perception of spatial properties such as size, location, and shape [4]. Humans rely more on the visual than kinesthetic cues when the visual information conflicts with the haptic information for both spatial and stiffness information [3,5].

In real life, familiar objects can usually be identified haptically (without vision) with virtually no error within a period of 1-2 seconds [1]. Object exploration tends to begin with general-purpose procedures that provide coarse information about the object and lead to specialized procedures to look for distinctive features of the unknown object [1]. Simplifying the haptic stimuli to a point requires more exploration time to acquire a complex object's feature information.

Given the complexity of the human haptic sensory system, simulating touch in a virtual environment is likely to be difficult. During haptic application demonstrations, new users struggle to learn the use of haptic device and consequently have harder time performing specific application tasks. Touch&Tell was developed to teach the use of the device first, thus preparing the user for more complex application tasks.

2. Touch&Tell

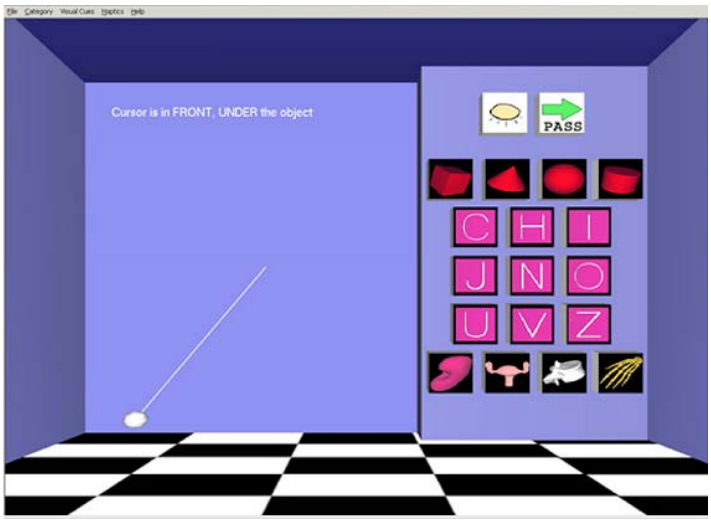

Figure 1. Touch & Tell game.

Initially, the user is shown how to move the haptic device to touch some simple 3D objects. After gaining some experience, the user is allowed to explore an object that is not displayed graphically. Separating the haptic and visual component requires the user to rely on the sense of touch to identify the object in question.

The device is used to touch an extendable library of objects, which is broken down into three groups of increasing complexity. These groups consist of solid geometric shapes, engraved alphabet blocks, and anatomical structures, Figure 1. As users explore the hidden object, they look for key characteristics that can be used for identification. For example, a cylinder would be identified if the object had a curved side and a flat top and bottom. The identification process forces controlled device movements in order to search for object's characteristics and thus refines user's control of the device. The geometric shapes help in general discrimination, alphabet objects require fine discrimination, and anatomical structures require both general and fine discrimination. In order to be successful, the user has to develop a strategy for object recognition.

An image key, shown in Figure 1, is provided to the user as a visual reference to aid in identifying the hidden object. After identifying the object, the user can press the corresponding haptic button on the image key to identify the object. A correct answer causes a bell sound, displays the hidden object, and then brings up the next hidden object. An incorrect decision causes a buzzer to sound and allows the user to make another guess.

2.1 Visual and audio cues

Optional visual and audio cues are included to help locate and stay in contact with the hidden object. The first visual cue is a line directing the proxy's graphical representation towards the center of the object. The second visual cue is a directional indication that tells the user whether the cursor is in front, behind, above, below, left, and/or right of the object. The combinations of the cues work well to reduce the level of difficulty for locating the object.

Once contact with an object is made the cursor turns green and a sound is played. When contact is lost, the cursor returns to the initial white color and another sound is played. These cues help the user to stay in contact with the object.

2.2 Changing the level of difficulty

An experienced user can choose to manipulate the objects (e.g. rotate, translate, and scale) and work at a more difficult level without the cues. There are other variable modes such as where the object category is known/unknown and the object is hard/soft. Using combinations of the different modes allows for increasing/decreasing the difficulty level of the task. For example, knowing the object's category reduces the number of possible choices and decreasing the stiffness of the object causes surface details to become less apparent. Other modes can be incorporated as needed for different experiments.

2.3 Data collection

In order to keep track of who is playing the game, the user must log in. All of the user's guesses are logged with a timestamp to determine the number of errors that occurred and the amount of time taken while identifying the unknown object. Two different timings are recorded. The first time is how long it takes to locate the object and the second is how long it takes to identify the object after the initial contact. Separation of these two times allows for a more accurate reading of how long the user actually explores the object in order to identify it. Touch&Tell also records contact information to keep track of the number of times the user gained/lost contact with the object. This helps in determining the ease of following the surface of the 3D object. Changes in strategy and user's difficulty level are also recorded. Based on a set of weighted parameters, the user's achieved level is calculated and stored.

3. Discussion

Although Touch&Tell is discussed in this paper as a tool for instructing users how to use the Phantom, it has many more potential applications. Touch&Tell can be a haptic research tool. Through various experiments, researchers trying to understand what roles the physiological and psychological factors (including vision) play for the sense of touch would benefit from this game. Other experiments, such as one performed by other researchers [6,7] to measure the Just Noticeable difference (JND) for force feedback, can be carried out. Researchers interested in the finer surface features [8] may be able to run experiments and see how effective the fine surface features are for identifying the objects. Experiments with Touch&Tell are expected to quantify how effective the sense of touch through a point-based device is for identifying 3D objects. This type of study will be beneficial for applications for visually impaired users.

4. Conclusions

Exploring a 3D environment with a point-based haptic device causes problems for new users. The Touch&Tell game teaches how to use the Phantom and how to control device movements in order to identify hidden objects by touch. Users utilize both general and fine discrimination in order to locate key characteristics of hidden objects for identification. This fun and challenging game is working as an effective tool to introduce haptics to new users.

Touch&Tell's data collection capabilities also make it possible to use it as a haptic research tool. Depending on the specific aim, Touch&Tell can be tailored to record different types of information.

A version of Touch&Tell called "ifeelit" is displayed as an exhibit in the Future Technology Center that Telecom Italia has created in Venice Italy.

5. Acknowledgements

The authors would like to thank Dr. Parvati Dev of the SUMMIT group at Stanford University and Dr. Mike Ackerman of NLM for their support; especially Dr. Leroy Heinrichs for being very enthusiastic and providing many great suggestions. This work was supported in part by the Surgery Department of the Texas Tech Health Sciences Center and the State of Texas Advanced Research Project Grant 003644-0117.

6. References

1. Klatzky, R., "Haptic Perception", MIT Cognet, http://cognet.mit.edu/MITECS/Entry/klatzky.html.
2. Burdea, G. C., *Force and Touch Feedback for Virtual Reality*, New York: Jon Wiley & Sons, INC., 1996.
3. Wu, W-C., Basdogan , C., Srinivasan, M. A., "Visual, Haptic, and Bimodal Perception of Size and Stiffness in Virtual Environments", *ASME Dynamic Systems and Control Division*, (DSC-Vol.67): 19-26, 1999.
4. Heller, M. A., Schiff, W., "The psychology of Touch", 1991.
5. Durfee, W. K., Hendrix C. M., Cheng, P., Varughese, G., "Influence of Haptic and Visual Displays on the Estimation of Virtual Environment Stiffness", *ASME Dynamic Systems and Control Division*, (DSC-Vol. 61): 139-144, 1997.
6. Srinivasan, M. A., Chen, J-s., "Human Performance in Controlling Normal Forces of Contact with Rigid Objects", *ASME Advances in Robotics, Mechatronics, and Haptic Interfaces*, (DSC-Vol. 49): 119-125, 1993.
7. Allin, S., Matsuoka, Y., Klatzky, R., "Measuring Just Noticeable Differences for Haptic Force Feedback: Implications for Rehabilitaion", *Proceedings of the 10th Symp. On Haptic Interfaces For Virtual Envir. & Teleoperator Systs. (Haptics '02)*, March 24-25, 2002.
8. West, A. M., Cutkosky, M. R., "Detection of Real and Virtual Fine Surface Features With a Haptic Interface and Stylus", *ASME Dynamic Systems and Control Division*, (DSC-Vol. 61): 159-166, 1997.

Using Haptics Interaction in Bioinformatics Application

Arthurine Breckenridge
arbreck@sandia.gov

Ben Hamlet
bhamlet@nmt.edu

Sandia National Laboratories, New Mexico is continuing to research 3D user interfaces specifically with the addition of haptics, the sense of touch. Our project is researching computational and collaboration tools independent of content area. However, applying the work to specific problems is very beneficial guide to our development. The application specific content focus of this paper is a bioinformatics application.

Bioinformatics is the combination of computer science and biology (the youngest of the natural sciences). Sequence-structure-function prediction refers to the idea that, given a molecule's sequence identity, we would like to predict its three-dimensional structure and, from that structure, infer its molecular function. Researchers have uncovered reasonable evidence indicating that a protein's structure approximately determines its molecular functions, such as catalysis, DNA binding, and cell component binding.

Many results in experimental biology first appear in image form – a photo of an organism, cells, gels, or micro-array scans. As the quantity of these results accelerates, automatic extraction features and meaning from experimental images becomes critical. At the other end of the data pipeline, naïve 2D or 3D visualizations alone are inadequate for exploring bioinformatics data. Biologists need a visual environment that facilitates exploring high-dimensional data dependent on many parameters. Therefore, visualization of bioinformatics is a vast field. Our specific focus is highlighted in special interest box: intron/exon splicing.

Basic Simulation Elements

The two basic elements of a computer haptics simulation are graphics and force that actually work independently of each other. This divides the simulation into two distinct parts, each of which present unique problems that can be addressed in independent and different ways. Although a number of factors go into making a successful simulation, the success of ones application will ultimately rely on good refresh rates (i.e., +30 Hz for graphics and 1000 Hz for force feedback). Both graphics and haptics interaction can apply to the application content and the graphic/haptic user interface. The application will use e-Touch ™ originally developed within Sandia National Laboratories, New Mexico and now licensed by Novint Technologies, Inc.

Graphic/Haptic User Interface

Force feedback has been implemented into the e-Touch ™ user interface. The basic elements are documented on the www.novint.com website. Features such as craft navigation and menus with touch sensations are standard. In addition, this application has added 3D box selection of elements and magnetic pointer to start location. The start location pointer can be used as intelligent alignment tool.

Graphics Generation

A number of approaches are available for rendering molecular graphics. One is a simple shape like line, sphere, etc. Although simple to code, as the number of primitives increase, the graphics refresh rate decreases substantially. Quick fixes are to sort the data by atom type to avoid graphics transition states like color and material properties. Keep in mind that data pre-processing is highly desirable for all haptics simulations due to the relentless refresh rates that such an application demands. Another approach is to calculate which surfaces are hidden. Although removing hidden surfaces is substantially

faster than drawing them, when dealing with a large number of atoms, it can still exceed the haptics servo loop requirements. Of course, the classic solution is to draw different levels of details based on screen position, user selection via graphical user interface, and spatial decomposition

Another graphical method used that seems promising is to fool the eye that it is seeing a 3D object since the sense of touch sends another signal that the object is actually has

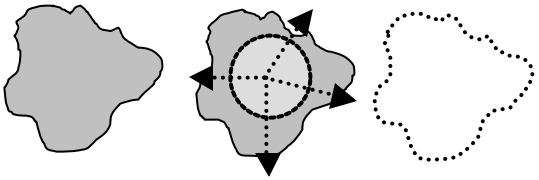

Figure 1. Circular Vector Field

shape (i.e., round atom). In this method, one determines the shape and contour of the protein's surface within a given level of detail and creates a graphics triangular surface mesh. For example, knowing the center location of each atom, the furthest sphere intercept along these vectors can be calculated (Figure 1).

Each point of intersection will then be used as a vertex in creating a surface mesh of triangles. The surface mesh can be rendered on a PC cluster since it requires an entire program to simply plot it. Once the coordinates of the surface points have been calculated, there is still a substantial amount of compute cycles needed to create a texture map for the protein and define the shadowing of the polygons. If it seems complex, this graphics solution is actually beneficial to maintain the needed haptics refresh rates and scale linearly.

Force Calculations

Computer haptics using the PHANToM is basically surface forces that act like springs, where the force exerted is equal to a constant times the depth of penetration, F= kx. Because force magnitude is dependent on the depth of penetration, when one touches two spheres at the same time but does not penetrate their surfaces an equal distance, one surface will exert a greater force than the other (Figure 2).

All of this takes place at 1000 Hz, which causes a high frequency oscillation of the PHANToM that can be felt or even heard in many circumstances. Force buzzing is simply where the PHANToM vibrates because of an inconsistent or unstable stream of forces is being fed to it. Object springiness can cause this because when two identical spheres push on the cursor, and the first sphere pushes harder than the second, the cursor will penetrate the second sphere more deeply, inducing a strong force in that sphere and so on. There are a number of solutions to this problem, many of which are not simple and may have undesirable side effects. The easiest way to eliminate buzzing, at least to the extend of audible recognition, is to vary the spring constant of the surface forces proportionally to the number of objects being touched. For example, if one begins to touch one atom, you will feel the maximum force. Then if one moves the cursor to touch more atoms, the force will decrease (.05 to .1) and will increase as you touch less. If one's range of forces and the speed at which one ramps them is reasonable, this method can alleviate solid surface buzzing.

Another solution to force buzzing produced by overlapping objects is to not have overlapping objects. Much in the same way that a graphics mesh of a protein molecule can be generated, a force shell for a protein can be created. By pre-processing the normals, one no longer has

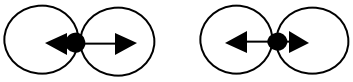

Figure 2. Force Buzzing

to worry about inter-object forces. A force shell will have the affect of making your protein completely impenetrable and may greatly reduce the value of one's simulation. Whereas the graphics shell still seemed 3D due to the force representation, this feature is lost if both graphics and force shells are used.

Special Interest: Intron/Exon splicing

The creation of life begins with a single cell that divides into two. And, why does this process sometimes malfunction, leading to defects, cancers, and other diseases? And, could this process sometimes function as planned and could potentially be used by terrorists. At the brink of the twenty-first century, there are 24 complete "draft" genomes available in public databases including that of the human genome. Though impressive information, the real challenge is transforming the torrent of raw data into biological knowledge. Thus, bioinformatics, the combination of biology and computer science is introduced. When asked what is the Holy Grail of bioinformatics, most researchers would answer the ultimate goal of a genome project is to determine the function and biological role for all of the genes of interest ideally in silico.

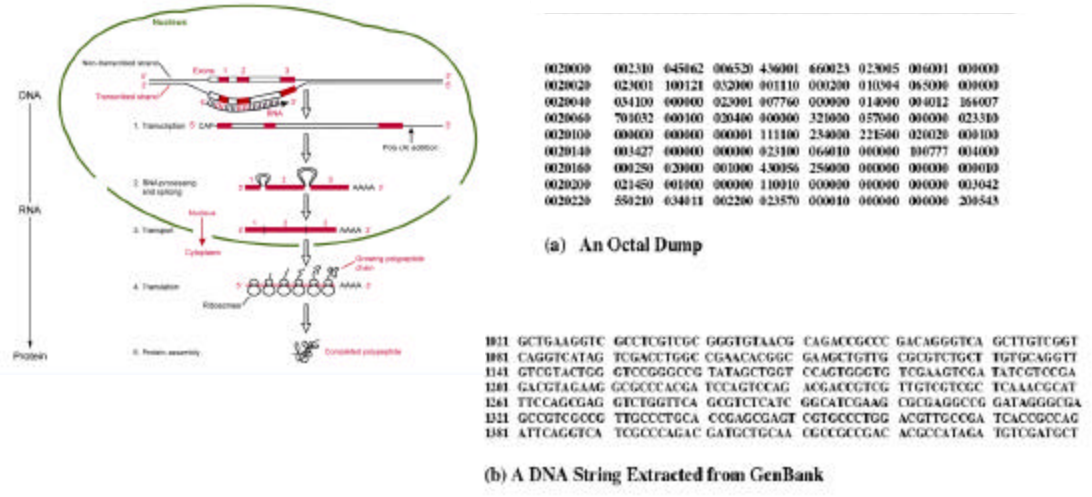

Figure A. Central Dogma as seen by a Biologist on the Right and a Computer Scientist on the Left

Though considerable progress has been made in understanding flow of information known as the "central dogma, Biology is the youngest of the natural sciences. Figure A.a) shows an octal dump of an assembly language code, which once upon a time ubiquitous in debugging computer programs, which nowadays become a rarity. Figure A.b) show a portion of the GenBank database presenting parts of DNA containing the letters A,C,G,T suggesting a quaternary kind of dump. Hopefully, this important but initial analysis will also become a rarity, as we understand more structural dynamics.

Because of the interdependent flow of information represented by the central dogma, one can begin discussion of the molecular expression of genetics of gene expression at any of its three informational levels: DNA, RNA, or protein. In 1953, James Watson and Francis Crick deduced the secondary structure of DNA. This was one of the most important biological advances, since it led to an understanding of the relationship of the DNA structure to its function, particularly to the way it was replicated. Scientists now are beginning to understand the process of copying DNA into RNA called transcription. The perceived role of RNA has changed from a passive messenger of information and scaffold for proteins to a central and active role in the functioning of the cell. To understand how a specific RNA molecule operates, its functional structure need to be better understood. It may be deduced from (costly and difficult to obtain) X-ray diffraction or NMR data only for short RNAs. In most cases, however, only the single RNA sequence (the primary structure) without further information regarding its functional form is available.

The secondary structure differs from the Watson-Crick DNA secondary structure in that it is generally single-stranded. When left in its environment, this molecule will fold itself into its secondary structure by creating base-pairs (an A with a U, a C with a G, or even a U with a G). Scientists are now deducing the secondary structure of RNA by using difference equations and dynamic programming. In 1977, it was determined that genes of higher organisms did not follow the simplest "central dogma" model. Instead, few genes exist as continuous coding sequences. Rather, one or more non-coding regions interrupt the vast majority of genes. These intervening sequences, called introns, are initially transcribed into pre-mRNA in the nucleus but are not present in the mature mRNA in the cytoplasm. Introns alternate with coding sequences, or exons, that ultimately encodes the amino acid sequence of the proteins. The portions corresponding to introns are removed, and the segments corresponding to exons are spliced together. The production of mRNA has to occur in the correct amount, in the correct place, and at the correct time during development or during the cell cycle. Each of the steps in this complex pathway is prone to error, and mutations that interfere with the individual steps have been implicated in a number of inherited genetic disorders.

Considering human genes contain ten times more intronic than exonic sequence, this weakness in the understanding of higher eukaryotes genetics is becoming increasingly apparent. Recent bioinformatic studies suggest that at least one third of human genes are alternatively spliced. Intron's evolutionary functionality is in much debate. However, there is agreement that the splicing sequences are neither strong nor unique enough signals since the splice sequence can be found in other parts of the mRNA. In the context of computer prediction of exon boundaries based only on primary sequences, this makes the task quite difficult. Therefore, what factors in the nucleus may also facilitate identification of sites? Shown in Figure A, this proposal is interested in process for detecting nuclear introns. Soon after discovery of splicing it became evident that this process also occurred in lower eukaryotes like yeast. Because the chemical mechanism of splicing and many factors involved in the splicing process are conserved from yeast to man, the budding yeast, Saccharomyces cerevisiae, has become an important model systems for the analysis of splicing. A comprehensive study of splicing in vivo of yeast (Spingola, 1999) concluded that results show that correct prediction of introns remains a significant barrier to understanding the structure, function and coding capacity of eukaryotic genomes, even in a supposedly simple system like yeast. As complete eukaryotic genome sequences become available, better methods for predicting RNA splicing signals in raw sequence will be necessary in order to discover genes and predict their expression.

Why study secondary structure? One of the major problems facing computational biology is the fact that sequence information is available in far greater quantities than information about the three-dimensional structure. While the prediction of 3D RNA structures is unfeasible at present, the prediction of secondary structures is in principle tractable. In RNA the secondary structure elements are significantly more stable and form faster than the tertiary interactions. Thus, a separation of an RNA folding model into secondary (properly nested base pairs), and tertiary (non-planar nucleotide contacts) seems feasible. Determining the secondary structure of an RNA molecule is widely seen as a first step towards understanding its biological function.

Spingola, M, Grate, L., Haussler, D., and Ares, M., "Genome-wide bioinformatic and molecular analysis of introns in Saccharomyces cerevisiae", *RNA*, 5:221-234, 1999.

Virtual Haptic Validation for Service Manual Generation

Christopher Volpe
volpecr@research.ge.com

Russell Blue
blue@research.ge.com

GE Global Research Center

Abstract

We've designed and implemented a virtual validation system using haptics as well as other typical Virtual Reality equipment (head-mounted display, data-glove, and tracking hardware) for validating maintenance procedures and doing maintenance studies. The system uses a compact, sampled data representation to handle complicated geometric environments. Thus, the rate of collision detection is independent of the number of polygons in the source data. Also key to the system is an independent representation of the parts in the environment, allowing a random-access removal order. This allows a maintainer to validate a sequence of steps in a maintenance task, or instead to determine the sequence of steps defining the maintenance task. The validation system is part of a larger system (Service Manual Generation) to produce maintenance procedures in an automated fashion.

Introduction

Maintainability analysis involves analyzing the design of a system to ensure maintenance actions can be performed quickly, safely and accurately. Complex systems such as aircraft, power systems and appliances require timely, reliable and accurate documentation to support maintenance. As a result of such complexities, technical manuals with textual descriptions and exploded views of equipment describing, for example, how to remove, inspect, repair and install parts, are required to keep many complex systems in working order. The documentation is an integral part of the maintenance process. Thus, a thorough maintainability analysis should include the analysis of the product design and documentation working together. Unfortunately, maintenance manuals are typically one of the last items developed and delivered in a complex system [1,2].

To address this problem, a research effort was begun under contract with the Air Force Research Lab's Human Effectiveness Directorate [3] to investigate an automated approach to maintenance manual development. The three year program, entitled "Service Manual Generation" (SMG) [2,4,5], was started in 2001 and consists of three main components: Sequence Generation (SG), Task Generation (TG), and Virtual Validation (VV). The SG component is responsible for analyzing the CAD geometry of the relevant parts, producing an exploded view of the assembly, and determining a sequence of part removals for the purpose of maintaining a particular part, such as a "Line Removable Unit" (LRU) of an aircraft engine. The TG component takes the output of the SG component, and generates content for a maintenance manual authoring system, including the human-readable steps to perform the task, and Warnings, Notes, & Cautions associated with any of the steps. The final component, VV, has two main responsibilities. First, it enables the validation of the removal sequence produced by the SG component, to ensure that the sequence itself is feasible by performing the tasks after being given a visual depiction of the procedure. Second, it allows the user to verify the instructions produced by the TG component to ensure that the instructions adequately describe the task.

The VV component combines a number of virtual-reality technologies in an effort to provide a realistic environment for determining the validity of the steps within the sequence. These technologies include a stereo Helmet-Mounted Display (HMD) to immerse the user in the environment, a data-glove for modeling the hand in the removal process, a tracking system for mapping the real-world positions of the HMD and data glove to positions in the virtual environment, and a 6DOF SensAble ™ PHANToM™ for enabling realistic human interaction with the virtual environment (Figure 1). It is the use of haptics in this virtual environment that is the main topic of this paper.

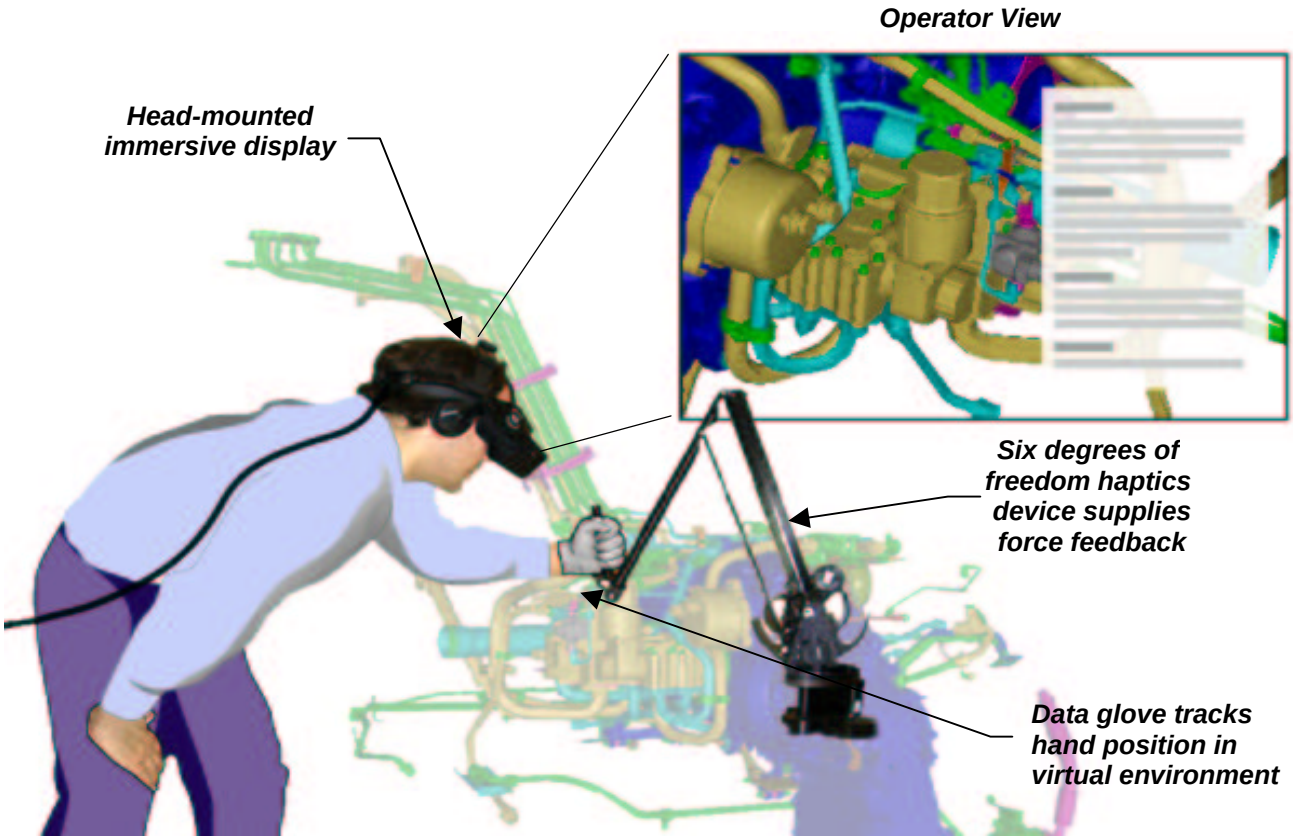

Figure 1: VV system

Haptics for Virtual Validation

GE Global Research has been working for some time on the use of haptics for maintainability analysis. Initial investigation for this task began in 1998, when we developed a collision technique based on using points sampled from the surface of a moving part, and a volumetric grid containing proximity/penetration information for the non-moving parts, similar to the technique developed around the same time by Boeing [6]. In the fall of 1999, we delivered a prototype Haptic Path Planner tool to the Joint Strike Fighter (JSF) military engine program at GE Aircraft Engines. This work provided the foundation for the haptic component of the Virtual Validation system in SMG. For Virtual Validation, we needed to enhance our approach to allow for parts to be selected in a random (user-determined) order as the moving part for removal. This allows multiple parts to be haptically removed in rapid succession. In the paragraphs below, we describe in some detail how we represent data in our haptic environment in a way that allows for multiple part removal, and how we perform fast collision detection, independent of the polygonal complexity of the models used in the environment.

Data Representation

There are three main components to our haptic representation. We describe each of these components in turn.

Surface Points. For each part that participates in the removal process, we generate a set of points that lie on the surface of the part which are evenly spaced and of sufficient density to adequately represent the part at a desired resolution. The surface points for a given part are used only when that part is the “moving part” in the environment.

Penetration Map. For each part relevant to the maintenance task, whether it is one to be removed or not, we generate a fine-resolution volumetric grid called the "penetration map", which contains proximity (penetration) information for points that lie outside (inside) the surface of the part, along with surface normals corresponding to the surface polygon nearest to the given point in the volumetric grid. In order to conserve space, this grid is arranged in a two level hierarchy in which a 4x4x4 block of grid points are grouped together to form a "brick". If all the grid points within a brick are greater than some maximum distance outside the part, or greater than some maximum distance inside the part, then the brick consists of a single value identifying that state. Otherwise, the brick is broken down into the 64 grid points with detailed penetration and surface normal information.

Part Map. The collection of (possibly overlapping) penetration maps of all the relevant parts defines a volume of space consisting of the union of all the volumes of the individual penetration maps. This combined volume is subdivided at a coarser resolution than the individual penetration maps to produce a grid called the "part map". This part map contains, at each grid location, a list of which parts are relevant within the space represented by the given part map grid location (Figure 2).

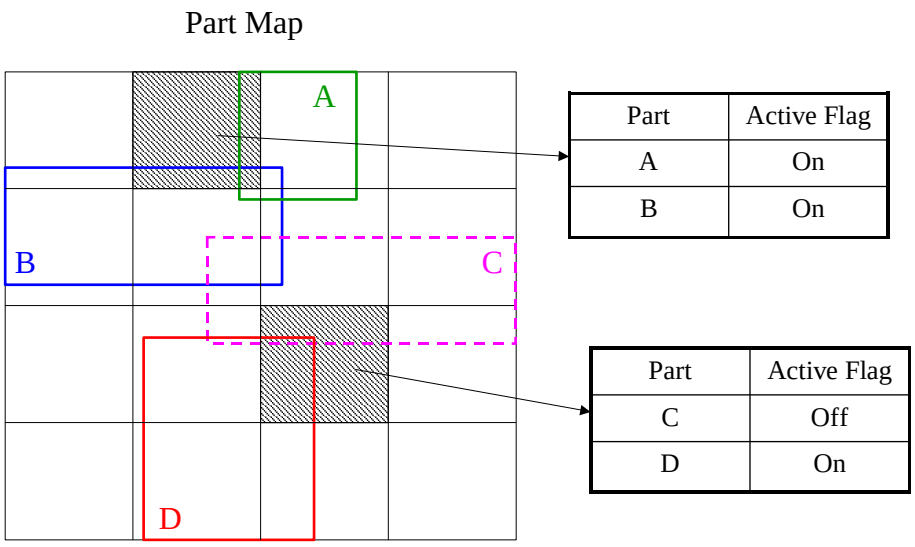

Figure 2: Part Map example

Fast Collision Detection

At each iteration of the haptic collision detection process, the sampled points of the moving part are transformed according to the moving part's position and orientation in the virtual environment. The resulting location is indexed into the part map to determine which parts are possibly in contact with the given point. Since any one of these parts may have been previously removed in an earlier step of the removal sequence, each part has associated with it an "active" status flag, indicating whether or not it participates in collisions with the moving part (Figure 2).

For each active part in the part list at the current sampled point's location, we conduct a penetration test with that part's penetration map. Based on the sampled point's location, we look up the eight surrounding grid points in the penetration map and tri-linearly interpolate both the proximity/penetration distance and the associated surface normals. We keep track of whichever part the sampled point penetrates the most, and use that part's interpolated penetration distance and surface normal to calculate a collision response.

In order to take advantage of a sufficiently dense set of sampled points without wasting precious CPU cycles on points that have no chance of being in collision, we take advantage of the temporal coherence of the moving part through a technique called "skip counts". Each time a collision test is performed on a sampled point, we determine,

based on the distance of that point from the nearest surface and the maximum speed with which we allow a part to move, the minimum number of iterations that must pass before that point can possibly collide with any surface. This number becomes the "skip count" for the given point. At each iteration of the haptic loop, we decrement the skip count of each point for which the skip-count is non-zero, and perform the collision test only for those points whose skip-count is zero.

Current Research Issues and Future Tasks

In order to make our system more robust, we are currently investigating a variety of techniques for dealing with frequently occurring pathological situations in real-world data. One such situation occurs when the part has regions where the thickness is on the order of our sampling resolution. We frequently see this case in aircraft engine data for hollow tubes where the wall of the tube is thin. When this occurs, adjacent penetration map elements may reside on either side of the wall, resulting in the ability to pass through a wall without ever landing within the interior material, and thus failing to detect the penetration. The result is that the part can erroneously move through some regions of the penetration maps. We are currently evaluating several candidate solutions to this problem.

In order to accurately model the affects of the user's hand in the environment, we need to implement the ability to allow both a sampled point model of the data glove and the moving part to participate in the collision response simultaneously. The colliding hand points will affect the haptic collision response in the same manner as the sampled points on the moving part. Furthermore, we can scale the hand model to perform human factor analyses of a task.

A similar needed interaction within a maintenance environment is that of tools, which must fit snugly over certain parts without being repelled, and which must be kinematically constrained to move with fewer degrees of freedom. This ensures that not only can a part be removed, but also that it can be removed using the tool designated for the task.

References

[1] E. Sanchez, S. Winning, E.S. Boyle. "Automated Support for Maintenance Technical Manuals", Air Force Armstrong Laboratory Technical Paper (AL/HR-TP-1997-0051).

[2] J. Wampler, R. Blue, C.R. Volpe, P. Rondot. "A Virtual Approach to Maintenance Manual Development", Concurrent Engineering - Research and Applications, Proceedings of the Ninth ISPE International Conference on Concurrent Engineering, 27-31 July 2002, Cranfield, United Kingdom, 2002.

[3] United States Air Force, Air Force Research Laboratory Contract Number F33615-01-2-6000.

[4] J. Wampler, R. Blue, C.R. Volpe, P. Rondot. "Service Manual Generation: An Automated Approach to Maintenance Manual Development", Proceedings of the 16th Human Factors in Aviation Maintenance Symposium, April 2-4, 2002, San Francisco, CA., 2002.

[5] J. Wampler, J. Bruno, R. Blue, L. Hoebel, "Integrating Maintainability and Data Development", Proceedings of the Annual Reliability and Maintainability Symposium, January 27-30, 2003, Tamp Bay, FL., 2003.

[6] W.A. McNeely, K.D. Puterbaugh, J.J Troy. "Six Degree of-Freedom Haptic Rendering Using Voxel Sampling." Proceedings ACM SIGGRAPH 99 Conference, August 1999, Los Angeles, CA, pp. 401-408, 1999.

Keynote Speaker -- Francois Conti

Co-Founder, and Co-CEO, Force Dimension
Francois Conti received his MS in Electrical Engineering from The Swiss Federal Institute of Technology in Lausanne (EPFL). He joined the VRAI group in 1998 and was responsible for the development of an endoscopic surgery simulator, today commercialized by Xitact. In 1999, He participated in the designed of the DELTA Haptic Device and co-founded Force Dimension in 2000. In 2001, he joined the Robotics Laboratory at Stanford University where he is currently pursuing his Ph. D. in the field of soft tissue modeling and simulation for real-time applications.

Force Dimension

Force Dimension, a spin-off of EPFL founded in 2000, designs and manufactures high-end force feedback interfaces for research and industrial applications. Force Dimension has been active in the automotive and aerospace industry for augmented haptic teleoperation and telemanipulation applications.

Company Milestones:

2000	Force Dimension is founded and the 3-DOF DELTA Haptic Device is commercialized.
2001	The 6-DOF DELTA is introduced.
2001	Support of Novint's eTouch API.
2002	The Mini DELTA is presented at Eurohaptics for the first time.
2002	Support of the ReachIn API
2002	NanoFeel, a spin-off company from EPFL, is launched to commercialize the NanoManipulator: a force feedback and visual software interface, based on the DELTA Haptic Device, controlling in real-time an Atomic Force Microscope (AFM)

Invited Speaker – Michael Wallace

Can a hand-held 3D input device that doesn't employ force-feedback be considered truly haptic? Can such a device outperform force-feedback devices? Could it complement such devices? Global Haptics thinks all of the above, at least in part. The history of this nanoscopic company is short but full of highlights, including the defeat of a major computer company in an IP-type battle. Mike Wallace, founder and President of Global Haptics, gives a summary of the 'haptic orb' technology and compares it (as best he can) to other force-feedback solutions. He'll apply his perspectives as an award-winning sculptor and earth scientist to how those fields are approached by haptics. He follows with his perceptions on the common hurdles that all haptics companies face in reaching a mass market.